MASTER'S THESIS

Gradient Flow Based Matrix Joint Diagonalization for Independent Component Analysis

by Bijan Afsari
Advisor: Professor P. S. Krishnaprasad

MS 2004-4

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2004		

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Gradient Flow Based Matrix Joint Diagonalization for Independent Component Analysis	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,The Graduate School,2123 Lee Building,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**116**	

ABSTRACT

Title of Thesis: GRADIENT FLOW BASED MATRIX
JOINT DIAGONALIZATION FOR
INDEPENDENT COMPONENT ANALYSIS

Bijan Afsari, Master of Science, 2004

Thesis directed by: Professor P. S. Krishnaprasad
Department of Electrical and Computer Engineering

In this thesis, employing the theory of matrix Lie groups, we develop gradient based flows for the problem of Simultaneous or Joint Diagonalization (JD) of a set of symmetric matrices. This problem has applications in many fields especially in the field of Independent Component Analysis (ICA). We consider both orthogonal and non-orthogonal JD. We view the JD problem as minimization of a common quadric cost function on a matrix group. We derive gradient based flows together with suitable discretizations for minimization of this cost function on the Riemannian manifolds of $\mathrm{O}(n)$ and $\mathrm{GL}(n)$.

We use the developed JD methods to introduce a new class of ICA algorithms that sphere the data, however do not restrict the subsequent search for the un-mixing matrix to orthogonal matrices. These methods provide robust ICA algorithms in Gaussian noise by making effective use of both second and higher order statistics.

GRADIENT FLOW BASED MATRIX JOINT DIAGONALIZATION FOR INDEPENDENT COMPONENT ANALYSIS

by

Bijan Afsari

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science 2004

Advisory Committee:

Professor P. S. Krishnaprasad, Chairman
Professor S. Shamma
Professor J. Simon
Professor A. Tits

DEDICATION

This thesis is humbly dedicated to all great men and women who have suffered in the path to enlightenment and democracy in my country, IRAN.

ACKNOWLEDGEMENTS

I would like to thank my advisor Professor P.S. Krishnaprasad for all his guidance and inspiration.

I would also like to thank the advisory committee for their insightful comments, especially Professor Andre Tits for his careful corrections and suggestions. I am also indebted to all University of Maryland members for providing an excellent environment for my achievement.

This research was supported in part by Army Research Office under ODDR&E MURI01 Program Grant No. DAAD19-01-1-0465 to the Center for Communicating Networked Control Systems (through Boston University).

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Background

The problems of Independent Component Analysis (ICA) and Blind Source Separation (BSS) are emerging and appealing areas of research. Starting in mid 1980's in Signal Processing and Neuroscience communities, ICA and BSS fast became the research focus of many different research communities. In their simplest formulation, ICA and BSS refer to the problem of retrieving n unknown *independent* random sources mixed through a mixing matrix and observed by n sensors without having or using any prior information about the source distributions or the mixing matrix. More specifically, let A be a non-singular matrix and $\vec{\mathbf{s}}(t)$ a vector of random independent sources and $\vec{\mathbf{x}}(t) = A\vec{\mathbf{s}}(t)$ the observed signal. The problem is to estimate A (the *mixing matrix*) or a matrix B that we call the *un-mixing matrix*, such that $\vec{\hat{\mathbf{s}}}(t) = B\vec{\mathbf{x}}(t)$ is an estimate of the source signals, by just using the samples of the sensed signal $\vec{\mathbf{x}}(t)$. It is apt here to mention that the terms BSS and ICA are used almost interchangeably, but BSS which is more common in array processing literature usually refers to the case that the linearity and in-

dependence in the model "truly" hold, whereas ICA refers to the case that these two are just assumed. Without further description the beauty and ubiquity of the problem is obvious. It is a very natural and immediate problem in many different subjects: Bio-Signal processing, seismic signal processing, communication channels, array sensor processing, financial data analysis,... all deal with problems that can be formulated in one or the other way as this problem. Till the 80's this problem was considered without enough attention paid to the word "independent", where"*independent*" was identified by or simplified to "*uncorrelated*". Although the independence in many cases is an inherent property related to the physics of the problems, it was ignored mainly because of computational difficulties, lack of knowledge about *Higher Order Statistics (HOS)* at least among engineers, and the dominance of *Gaussian* signal model assumption among researchers. In the mid 80's it was noticed that by resorting to higher order statistics or non-linear functions in measuring "independence" it is possible to achieve results that are impossible to have by just using second order statistics or linear correlation. Needless to say ICA's predecessor Principal Component Analysis (PCA) is based on eigen decomposition of the data's covariance matrix.

Since then a huge amount of research has been dedicated to this problem, its different versions and extensions. It is easy to see the influence of various fields on the ICA/BSS literature, just to mention: Statistics, Statistical Signal Processing, Neural Networks, Optimization Theory, Nonlinear Dynamical Systems, Differential and Information Geometry, Neuroscience. As a matter of fact, in this thesis we pursue a path that encompasses elements of Statistics, Differential Geometry, Dynamical Systems and Optimization theory.

1.2 Outlines and Contributions

In this thesis we develop new methods to solve a problem known as " Matrix Joint Diagonalization (JD)" which has applications in the BSS/ICA problem in addition to other contexts. The Joint Diagonalization problem in one of its forms is simply: for a set of symmetric matrices $\{C_i\}_{i=1}^N$ find a non-singular matrix B such that BC_iB^T's are "as close to diagonal as possible" (for more detail see Chapter 3). This, for example, can be encountered in the case that all C_i's are believed theoretically to be of the form $C_i = A\Lambda_i A^T$ for some common non-singular matrix A and diagonal Λ_i, but because C_i's sample estimates are used this can hold only approximately. Hence the idea would be to find a matrix B that diagonalizes all of them simultaneously, as much as possible. Our approach to this problem is to introduce suitable cost functions for the problem and use ideas from differential geometry to derive gradient based *matrix ordinary differential equations* (ODEs) on certain Riemannian manifolds such that the ODEs solution can yield (or converge to) a joint diagonalizer. In deriving ODEs for optimization purposes we mainly follow [Brockett] and [Helmke]. We also give some discretization schemes for these (ODEs) with an eye to keeping the answers on the underlying manifolds.

In the context of ICA/BSS problem the matrices to be jointly diagonalized can be "cumulant slices" [Cardoso1, Yeredor]. Applying the developed methods to the Joint Diagonalization of cumulant slices, we devise algorithms for the ICA/BSS problem, that are effective for ICA in the presence of *Gaussian noise*.

In Chapter 2, we introduce and formulate the ICA/BSS problem, and explain some of the basic definitions about it. We give the fundamental theoretical results about the issue of identifiability in the ICA/BSS problem. We also introduce measures of independence, definitions and some useful results about cumulants.

The group structure of the ICA/BSS problem is a very important concept that is introduced there. A short survey of ICA problems is also presented.

In Chapter 3, we elaborate more on the criteria of Joint Diagonalization of cumulant matrices. We consider some of the commonly used cost functions and their validity, we introduce some new cost functions, as well.

In Chapter 4, we provide very briefly the required material from the theory of matrix Lie groups. Our treatment will be very short and informal.

In Chapter 5, following [Brockett] and [Helmke], we introduce a gradient flow for joint diagonalization by orthogonal matrices, and consider their convergence properties. The Double Bracket [Helmke,Brockett] formulation of joint diagonalization is introduced there. We also discretize the orthogonal flow (using the Euler and Runge-Kutta methods) to give a gradient based version of the famous JADE [Cardoso1] algorithm for the BSS/ICA problem. We also give some numerical examples manifesting the performance of the developed algorithms.

In Chapter 6, the next contribution of this thesis, we develop flows for joint diagonalization by *non-orthogonal* matrices, and discretize them (using the Euler and Runge-Kutta methods). More specifically, we derive flows on the manifold of non-singular matrices $\mathrm{GL}(n)$ and matrices with unity determinant $\mathrm{SL}(n)$ for the JD problem. We will introduce methods to discretize them as well, so that the answer stays on the underlying manifold to a good extent.

In Chapter 7, we suggest a class of ICA/BSS algorithms based on the methods developed in Chapters 5 and 6. We introduce algorithms that *whiten* the data (using second order statistics, i.e correlations) in the first step and then search for *non-orthogonal un-mixing matrices* via joint diagonalization of a set of cumulant slices of the whitened data, hence resorting to the HOS of the data. This approach

is different from the existing non-orthogonal JD methods in the ICA/BSS context and also different from methods that use only HOS. In fact our method uses the benefits both of second order statistics that are robust and *cumulants* which are blind to Gaussian noise. In this chapter we also examine the actual performance of the developed methods by numerical simulations. Chapters 5,6 and 7 constitute the main contributions of this thesis.

In Chapter 8, we give a summary of the thesis as well as some suggestions for future work.

In the Appendix we have included some derivations that are rather lengthy to be included in main chapters or those that their omission do not harm the sequence of the subjects.

Chapter 2

Preliminaries about ICA/BSS

2.1 Problem Formulation

First we formulate the ICA/BSS problem in a general fashion and give different relevant assumptions. Consider

$$\vec{\mathbf{x}}(t) = A * \vec{\mathbf{s}}\,(t) + \vec{\mathbf{n}}(t) \tag{2.1}$$

where: t denotes continuous or discrete time, $\vec{\mathbf{s}}$ is an n dimensional random signal, $*$ indicates the linear convolution operation, $A = A(t, \tau)$ is an $m \times n$ convolution kernel or channel distortion matrix, $\vec{\mathbf{n}}(t)$ is an n-dimensional additive noise, and $\vec{\mathbf{x}}(t)$ is an m-dimensional observed signal. For many problems this model is quite realistic. It should be noted that because of the presence of the convolution operation in (2.1) the corresponding restoration or inverse problem is also referred to as Blind Deconvolution problem, as we will see with some additional assumptions this problem reduces to the standard BSS/ICA problem. We should also mention that in what is known as Nonlinear ICA, the assumption of linear mixture is relaxed, although that problem is much more difficult than the customary ICA formulation

presented here [Hyvarinen].

2.1.1 Some Possible Assumptions about the Model

Here we cite some different possible assumptions about each of the introduced variables:

1. **Assumptions about the Source Vector $\vec{\mathbf{s}}(t)$:**

   S1.The source $\vec{\mathbf{s}}$ is a vector of n independent stochastic processes,

   S2. $\vec{\mathbf{s}}$ is a *stationary* random process, with zero mean, with non-singular and finite correlation,

   $\sim$S2 . $\vec{\mathbf{s}}$ is a *non-stationary* random process,

   S3. Each component of $\vec{\mathbf{s}}$ is Independently Identically Distributed (i.i.d) in time,

   S4. In addition to S1,S2,S3, $\vec{\mathbf{s}}$ has *at most one* component with *Gaussian* distribution,

   S5.1. $n > m$, i.e more sources than sensors,

   S5.2. $m = n$, the same number of sources and sensors,

   S5.3. $n \leq m$, i.e more sensors than sources,

   S6. Sources are complex valued.

2. **Assumptions about the Mixing Channel $A(t, \tau)$:**

   A1. $A = A(t, \tau)$ is a time-invariant filter.

   A2. Impulse response $A = A(t, \tau)$ is instantaneous, i.e. $A_{ij}(t, \tau) = a_{ij}(t)\delta(t - (\tau + \tau_{ij}(t)))$, where $\delta(\tau)$ denotes the Dirac delta function or unit impulse in the discrete-time case.

   A3. Impulse response is memoryless, i.e. $\tau_{ij}(t) = 0$ and $A_{ij}(t, \tau) = a_{ij}(t)\delta(t - \tau)$.

A4. All assumptions A1-3 hold, that is the mixing process is a memoryless, instantaneous and time independent one, so the model (2.1) boils down to:

$$\vec{\mathbf{x}}(t) = \vec{\mathbf{z}} + \vec{\mathbf{n}} = A\vec{\mathbf{s}}(t) + \vec{\mathbf{n}}(t) \tag{2.2}$$

where we assume that A is full rank.

3. **Assumptions about the Noise Vector $\vec{\mathbf{n}}(t)$:**

   N1. Noise is *Gaussian*.

   ∼N1. Noise is not *Gaussian*.

   N2. Noise is stationary.

   N3. Noise covariance matrix is known.

   N4. Noise and signal vector $\vec{\mathbf{s}}$ are statistically independent.

Adopting each of these assumptions has significant impacts on solvability of the problem and as well as on the corresponding algorithms. For example condition ∼S2 leads to the problem of non-stationary BBS [Pham1], which interestingly is solvable by resorting to only second order statistics. As another example, under ∼N1 using higher order statistics (HOS) is not helpful because as well shall see in Section 2.4 HOS are blind only to Gaussian noise. By adopting N3 the estimation of the correlation matrix of $\vec{\mathbf{z}}$ will be "less biased" and the subsequent ICA algorithm will be easier.(see Section 7.1 for more details)

2.1.2 What is the ICA Problem?

Consider the model (2.2) with extra assumptions S1-4, S5.3 and N1,2,4, then the standard ICA problem can be stated as follows (we can assume complex data and channel but we avoid it). By just observing the realizations of the received signal

$\vec{\mathbf{x}}$:

F1. Estimate A the mixing matrix, or

F2. Find a matrix B such that $\vec{\mathbf{y}} = B\vec{\mathbf{x}}$ is an estimate of all or a subset of sources, or

F3. Find a matrix B such that the elements of $\vec{\mathbf{y}} = B\vec{\mathbf{x}}$ are as independent as possible.

In general these statements are not equivalent. Especially F3 is interesting because it relates the restoration of the samples of the data to the statistical independence of the restored signals. As we will see in the next section the key idea in solving this problem is *restoring the independence*.

2.1.3 Identifiability Conditions

Some immediate questions are: is it possible to restore $\vec{\mathbf{s}}$ exactly?, are there any ambiguity in the restoration?, under what conditions is the model identifiable?. To answer these questions we simplify the model another step and consider the noiseless model:

$$\vec{\mathbf{x}} = A\vec{\mathbf{s}} \tag{2.3}$$

Due to the multiplicative form of the equation, obviously any *blind* restoration will be up to a scaling factor unless we specify something about the power of the signal or elements of A. On the other hand a priori we have no information about the ordering of the elements in $\vec{\mathbf{s}}$, thus again any *blind* restoration will be up to a permutation of the sources, as well. It turns out that under some extra conditions these are the only two ambiguities or indeterminacies. We state without proof a theorem from Comon's paper [Comon 1] in a rephrased manner. Before that we give this definition:

Definition 2.1 *Two vectors or matrices A and B are called essentially the same or essentially equal if there is an invertible diagonal matrix Λ and a permutation matrix Π such that $A = \Lambda\Pi B$.*

Theorem 2.1 *[Comon 1] Suppose that the model $\vec{\mathbf{x}} = A\vec{\mathbf{s}}$ holds with conditions S4 and S5.3. If the elements of $\vec{\mathbf{y}} = B\vec{\mathbf{s}}$ are independent for some B, then BA is essentially diagonal, hence $\vec{\mathbf{y}}$ and $\vec{\mathbf{s}}$ are essentially the same.*

So this theorem states that in the case that $\vec{\mathbf{s}}$ is a random vector with at most one Gaussian component with some other mild conditions restoring independence and separating the sources are equivalent, and this process has an inherent ambiguity which is about the scale and order of the sources. This theorem is the theoretical core for almost all ICA/BSS algorithms. That is, all these algorithms introduce a criterion for independence and try to optimize it. Note that this theorem relates the BSS problem to the ICA problem, as well. The condition of having at most one component with Gaussian distribution is an important requirement, because for Gaussian vectors uncorrelatedness and independence are equivalent, and an uncorrelated Gaussian vector multiplied by an orthogonal matrix remains independent. Thus if there are more than one Gaussian components, then the ambiguity will be up to an orthogonal matrix which is not acceptable in a separation context.

Now if we consider the noisy case, we can argue that in the best case we can estimate B such that $BA = \Lambda\Pi$ and even $\vec{\mathbf{y}} = BA\vec{\mathbf{s}} + B\vec{\mathbf{n}}$ can have independent components but $\vec{\mathbf{y}}$ and $\vec{\mathbf{s}}$ are not essentially the same, that is through linear transformations we can not restore the sources, although we may be able to restore independence or find A. In fact resorting to higher order statistics, in Gaussian noise, we are able to find B such that $BA = \Lambda\Pi$, but that is not enough to restore the sources noiselessly.

In the next section we introduce measures of independence and their finite sample estimates.

2.2 Measures of Independence and Contrast Functions

We follow Cardoso's formulation as in Chapter 4 of [Haykin]. Let $\vec{\mathbf{x}}$ be an n-dimensional random vector with probability density function (p.d.f.) $f_{\mathbf{x}}(.)$ (we also write $\vec{\mathbf{x}} \sim f_{\mathbf{x}}(.)$) and let $\vec{\mathbf{x}}^p$ be its independentized version; that is $f_{\mathbf{x}^p}(.) = \prod_{i=1}^{n} f_{\mathbf{x}_i}(.)$. If $\vec{\mathbf{x}} \sim f_{\mathbf{x}}(.)$ and $\vec{\mathbf{s}} \sim f_{\mathbf{s}}(.)$ then the Kullback-Leibler (KL) divergence between random vectors $\vec{\mathbf{s}}$ and $\vec{\mathbf{x}}$ is defined as $D[\vec{\mathbf{s}} \parallel \vec{\mathbf{x}}] = \int f_{\mathbf{s}}(u) \log(\frac{f_{\mathbf{s}}(u)}{f_{\mathbf{x}}(u)})\, du$ and using the concavity of logarithm function it can be proved that $D[\vec{\mathbf{s}} \parallel \vec{\mathbf{x}}] = 0$ if and only if $f_{\mathbf{s}} = f_{\mathbf{x}}$ almost surely [Cover]. An important property of $D[. \parallel .]$ is that for any two n-dimensional random vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ and non-singular matrix A we have:

$$D[\vec{\mathbf{x}} \parallel \vec{\mathbf{y}}] = D[A\vec{\mathbf{x}} \parallel A\vec{\mathbf{y}}]$$

that is the KL divergence is invariant under one-to-one linear transformations.

The *mutual information* between the elements of $\vec{\mathbf{x}}$ is defined as $I[\vec{\mathbf{x}}] = D[\vec{\mathbf{x}} \parallel \vec{\mathbf{x}^p}]$. So $I[\vec{\mathbf{x}}] = 0$ if and only if the elements of $\vec{\mathbf{x}}$ are independent. In fact it can be shown that

$$I[\vec{\mathbf{x}}] = \min_{\vec{\mathbf{z}} \in \mathcal{P}} D[\vec{\mathbf{x}} \parallel \vec{\mathbf{z}}]$$

where $\mathcal{P}$ is the manifold [1] of random vectors with independent components and

[1] Here we use the term "manifold" in reference to an *infinite* dimensional manifold of probability densities whereas in the subsequent chapters we use this term to refer to finite dimensional manifolds. For rigorous definition of the former the reader is referred to [Amari 1].

this minimum is achieved for $\vec{\mathbf{z}} = \vec{\mathbf{x}}^p$, i.e. $\vec{\mathbf{x}}^p$ can be considered as the projection of $\vec{\mathbf{x}}$ onto $\mathcal{P}$ but obviously not in a usual sense because KL divergence is not a true metric. So we can rephrase the ICA problem in a compact way as:

$$\min_{B \in \mathbb{R}^{n \times m} \text{and full rank}} I[B\vec{\mathbf{x}}]$$

Obviously, this is not a very constructive way!, but it is an instructive one, that the goal is to minimize the output's mutual information. Later on we will see that the space of full rank $n \times m$ matrices is "too big" for this search, mainly because of the scale-permutation ambiguity in the ICA problem. Cardoso has shown that a maximum likelihood approach to ICA also leads to the same criterion [Cardoso3]. There are also other criteria, but they are somehow derived from mutual information [Haykin,Hyavarinen].

Mutual information is an example of a *contrast function*, a function in terms of an unknown parameter whose optimization (in this case its minimzation) with respect to that parameter gives independence or solves the ICA problem. For our purposes contrast functions are ICA cost functions. Obviously a good cost function is one that has only global optima and all the optimizers are essentially the same and give independence. So a good contrast function should be scale and permutation invariant as the mutual information is.

Evidently the problem with the mutual information is that it is a mathematical expectation and it depends on the p.d.f of $\vec{\mathbf{x}}$ which under the assumption of ICA is not known!. There are different ways to ameliorate this problem, among them is approximating the mutual information based on data samples and developing contrast functions accordingly. This is the approach we will follow.

2.3 Cumulants

We review the definition and properties of cumulants of random vectors. Intuitively cumulants are higher order correlations, so they can be considered as measures of independence. In some steps we follow [Porat] in notations. For an n-dimensional real random vector $\vec{\mathbf{x}}$ the moment generating function is defined as:

$$M_{\mathbf{x}}(\vec{\lambda}) = E\{\exp(\vec{\lambda}^T\vec{\mathbf{x}})\} \qquad ,\vec{\lambda} \in \mathrm{R}^n$$

The cumulant generating function then is defined as $C_{\mathbf{x}}(\vec{\lambda}) = \log M_{\mathbf{x}}(\vec{\lambda})$. Let $1 \leq i_1, i_2, ..., i_k \leq n$ then the k^{th} order cumulant of $\vec{\mathbf{x}}$ is defined as a k-way array whose element at the position $(i_1, i_2, ..., i_k)$ is:

$$Cum(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}) = Cum_{\mathbf{x}}(i_1, ..., i_k) = \left.\frac{\partial^k C_{\mathbf{x}}(\vec{\lambda})}{\partial\lambda_{i_1}...\partial\lambda_{i_k}}\right|_{\vec{\lambda}=0}$$

In the case of real valued sources, cumulants and moments both are permutation invariant. It can be proved that cumulants and moments can be derived from each other and in fact in practice cumulants are estimated via estimating the moments. For example the 4^{th} order cumulant for zero mean random variables, which we will use often, can be expressed in terms of moments as:

$$Cum(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = E\{\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4\} - E\{\mathbf{x}_1\mathbf{x}_2\}E\{\mathbf{x}_3\mathbf{x}_4\} - E\{\mathbf{x}_1\mathbf{x}_3\}E\{\mathbf{x}_2\mathbf{x}_4\}$$

$$-E\{\mathbf{x}_2\mathbf{x}_3\}E\{\mathbf{x}_1\mathbf{x}_4\}$$

In practice we ought to estimate the cumulants from sample data, this is usually based on sample averaging of expressions such as above, which in general is not robust to outliers. It can be shown that the higher the order of the estimated cumulant the larger the variance or error of the estimation would be. This is one of the main drawbacks of the cumulant based methods, which makes them

vulnerable in small sample sizes. There are other approaches and modifications to sample averaging method. For example one can filter out outliers, out of some distance from the mean. Methods such as this are costly to apply, and in fact it turns out that a huge portion of the computational cost in many cumulant based algorithms is estimating the cumulants rather than the computations afterwards.

Some of the properties of cumulants that are derived from the above definition are as follows:

C1. Cumulants are multilinear:

$$Cum(\mathbf{x}_{i_1}, ..., \alpha\mathbf{x} + \beta\mathbf{y}, ..., \mathbf{x}_{i_k}) = \alpha Cum(\mathbf{x}_{i_1}, ..., \mathbf{x}, ..., \mathbf{x}_{i_k}) + \beta Cum(\mathbf{x}_{i_1}, ..., \mathbf{y}, ..., \mathbf{x}_{i_k})$$

As a result, for $\vec{\mathbf{x}} = A\vec{\mathbf{s}}$ we have

$$Cum(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}) = \sum_{j_1 ... j_k} a_{i_1 j_1} ... a_{i_k j_k} Cum(\mathbf{s}_{j_1}, ..., \mathbf{s}_{j_k})$$

C2. If $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ are any two independent vectors then for cumulants of order k:

$$Cum_{\mathbf{x}+\mathbf{y}}(i_1, ..., i_k) = Cum_{\mathbf{x}}(i_1, ..., i_k) + Cum_{\mathbf{y}}(i_1, ..., i_k)$$

C3. If $\{\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}\}$ can be partitioned in two independent subsets then:

$$Cum(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}) = 0$$

Remark: If $i_1 = i_2 = ... = i_k = i$ then $Cum_{\mathbf{x}}(i, ..., i)$ is called the k^{th} order (auto)cumulant of the i^{th} component of $\vec{\mathbf{x}}$, otherwise $Cum_{\mathbf{x}}(i_1, ..., i_k)$ is called *cross cumulant*.

Corollary: If the components of $\vec{\mathbf{x}}$ are independent then all the cross cumulants of any order are zero.

C4. The Gaussian random vector is the only vector that all its cumulants of order $k > 2$ are zero.

By their multi-linearity, cumulants can be considered as tensors [McCullagh], and the ICA problem can be interpreted as tensor diagonalization. We can restate Comon's theorem as:

Corollary: In the standard ICA formulation (with assumptions in Theorem 2.1) $\vec{\mathbf{x}} = A_{m\times n}\vec{\mathbf{s}}$, if B is a matrix that makes the cumulant tensor of $\vec{\mathbf{y}} = B_{n\times m}\vec{\mathbf{x}}$ diagonal then $BA = \Lambda P$, where Λ is a diagonal and P is a permutation matrix. By a diagonal tensor we mean one with the property that its $i_1...i_k$ element is nonzero only if $i_1 = ... = i_k$.

In many cases we are interested in *cumulant matrix slices*, which we find from the cumulant tensor by fixing all but two of the indices. We denote a cumulant slice as: $Cum_{\mathbf{x}}(i_1, ..., i_{p-1}, :, i_{p+1}, ..., i_{q-1}, :, i_{q+1}..., i_k)$, notice the sign ":" which shows that index ranges over all its possible values. Obviously any cumulant slice of a real random vector is symmetric. Now we prove a lemma which is essential to the approach chosen in this thesis.

Notation: $\delta_{i_1,...,i_k,...,i_n}$ denotes the Kronecker delta and is equal to 1 if $i_1 = ... = i_n$ and is zero otherwise.

Lemma 2.1 *: Let $\vec{\mathbf{x}} = A_{m\times n}\vec{\mathbf{s}}$ and $\vec{\mathbf{s}}$ be an n-dimensional vector with independent components, then every cumulant slice(matrix) of $\vec{\mathbf{x}}$ is of the form*

$$Cum_{\mathbf{x}}(i_1, ..., i_{p-1}, :, i_{p+1}, ..., i_{q-1}, :, i_{q+1}..., i_k) = A\Lambda A^T$$

where Λ is a diagonal matrix of the form:

$$\Lambda = \begin{bmatrix} \lambda_{11}Cum_{\mathbf{s}}(1, ..., 1) & 0 & \cdots \\ 0 & \lambda_{22}Cum_{\mathbf{s}}(2, ..., 2) & \cdots \\ 0 & \ddots & 0 \\ \cdots & 0 & \lambda_{nn}Cum_{\mathbf{s}}(n, ..., n) \end{bmatrix}$$

and $\lambda_{jj} = \prod_{l=1,l\neq p,q}^{k} a_{i_l j}$.

Proof: Using the second part of C1 and the fact that because of the independence of the components of $\vec{\mathbf{s}}$, $Cum_{\mathbf{s}}(i_1, ..., i, .., i_k) = \delta_{i_1,...,i,...,i_k} Cum_{\mathbf{s}}(i, ..., i)$, we have:

$$Cum_{\mathbf{x}}(i_1, ..., i_{p-1}, i_p, i_{p+1}, ..., i_{q-1}, i_q, i_{q+1}..., i_k) =$$

$$\sum_{i=1}^{n} \prod_{l=1}^{k} a_{i_l i} Cum_{\mathbf{s}}(i, ..., i) = \sum_{i} a_{i_p i} a_{i_q i} (\prod_{l=1 l\neq p,q}^{k} a_{i_l i}) \, Cum_{\mathbf{s}}(i, ..., i) =$$

$$\sum_{i} a_{i_p i} a_{i_q i} \lambda_{ii} Cum_{\mathbf{s}}(i, ...i)$$

The last summation as i_p and i_q range between 0 and m can be written in the desired $A\Lambda A^T$ form. $\triangle$

Note that in the above lemma A is not restricted to be square. It is interesting to see that Λ depends on both source cumulants and elements of A for $k > 2$. In the special case of 2^{nd} order cumulant slice, i.e. covariance, Λ does not depend on the elements of A. This difference is important, as a covariance matrix is always positive semi-definite but a cumulant slice is not, necessarily. This lemma implies that all the cumulant slices of $\vec{\mathbf{x}}$ are diagonalizable (in a congruence manner) by the pseudo-inverse of A or in the case $m = n$ by A^{-1}. This observation is the basis for the ICA/BSS methods developed in thesis.

2.4 Cumulant Based ICA

As we mentioned before, mutual information is a measure of independence and cross cumulants show how much the variables are dependent on each other. So it would be interesting to see how these measures are related to each other.

2.4.1 Gaussian Manifold and the Negentropy

Here we follow Cardoso as in Chapter 4 of [Haykin]. Let $\mathcal{N}$ be the manifold of zero-mean Gaussian random vectors . It is easy to prove that if $\vec{\mathbf{n}} \in \mathcal{N}$ minimizes $D[\vec{\mathbf{x}} \parallel \vec{\mathbf{n}}]$ for finite covariance zero-mean random vector $\vec{\mathbf{x}}$ then $\vec{\mathbf{n}}]$ is a Gaussian vector with the same covariance as $\vec{\mathbf{x}}$. We denote this Gaussianized version of $\vec{\mathbf{x}}$ as $\vec{\mathbf{x}^n}$. The quantity $N[\vec{\mathbf{x}}] = D[\vec{\mathbf{x}} \parallel \vec{\mathbf{x}^n}]$ which measures the KL-distance of $\vec{\mathbf{x}}$ from $\mathcal{N}$ is called the Negentropy of $\vec{\mathbf{x}}$. We know that for any $\vec{\mathbf{n}} \in \mathcal{N}$, and any non-singular matrix A, $A\vec{\mathbf{n}} \in \mathcal{N}$, so using invariance of the KL divergence under 1-1 linear transformations we have:

$$N[\vec{\mathbf{x}}] = \min_{\vec{\mathbf{n}} \in \mathcal{N}} D[\vec{\mathbf{x}} \parallel \vec{\mathbf{n}}] = \min_{A\vec{\mathbf{n}} \in \mathcal{N}} D[A\vec{\mathbf{x}} \parallel A\vec{\mathbf{n}}] = \min_{\vec{\mathbf{n}} \in \mathcal{N}} D[A\vec{\mathbf{x}} \parallel \vec{\mathbf{n}}] = N[A\vec{\mathbf{x}}]$$

thus $N[\vec{\mathbf{x}}]$ is invariant under one-to-one linear transformations.

For a scalar random variable $\mathbf{z}$ with unit variance, $I[\mathbf{z}]$ can be approximated in terms of its cumulants as [Comon1]:

$$I[\mathbf{z}] \simeq \frac{1}{12}\kappa_3^2 + \frac{1}{48}\kappa_4^2 + \frac{7}{48}\kappa_3^4 - \frac{1}{8}\kappa_3^2\kappa_4 \tag{2.4}$$

where $\kappa_3 = Cum(\mathbf{z}, \mathbf{z}, \mathbf{z})$ and $\kappa_4 = Cum(\mathbf{z}, \mathbf{z}, \mathbf{z}, \mathbf{z})$. The proof of this is based on the approximation of the p.d.f. of z around the pdf of its gaussianized version, in an expansion known as Edgeworth expansion [McCullagh].

2.4.2 The Negentropy and Mutual Information

It is easy to show that $N[\vec{\mathbf{x}}]$ and $I[\vec{\mathbf{x}}]$ are related as

$$I[\vec{\mathbf{x}}] = N[\vec{\mathbf{x}}] - \sum_{i=1}^{n} N[\mathbf{x}_i] \; + \frac{1}{2}\log\big(\frac{\prod_{i=1}^{n} R_{ii}}{\det(R)}\big) \tag{2.5}$$

where R is the covariance matrix of $\vec{\mathbf{x}}$. By the Hadamard inequality [Marcus] about positive definite matrices, for a positive definite matrix R, $\log\big(\frac{\prod_{i=1}^{n} R_{ii}}{\det(R)}\big) \geq 0$, with

equality iff R is diagonal. In order to minimize $I[B\vec{\mathbf{x}}]$ with respect to B , we note that the first term in (2.5), i.e. $N[B\vec{\mathbf{x}}]$ is independent of B, so we need just to minimize the other two terms. In the sequel we show that a separate minimization of these two terms serves the goal of ICA.

2.4.3 Whitening the Data

By *whitening* or *sphering* the data we can minimize the last term in (2.5) and also confine the search for un-mixing matrix to the set of orthogonal matrices.
For the proof of the next theorem, we state a useful lemma first:

Lemma 2.2 *: Let $m \geq n$, and $P_{n\times m}$ and $Q_{n\times m}$ be full rank matrices. If $PP^T = QQ^T$ then P and Q are the same up to an orthogonal factor, that is there exists an orthogonal $U_{n\times n}$ such that $P = QU$ or equivalently $Q^\dagger P = U$, where $Q^\dagger$ is the right pseudo-inverse of Q.*

Proof: Let $P = V_1\Sigma_1 U_1^T$ and $Q = V_2\Sigma_2 U_2^T$ be the Economical Singular Value Decomposition (ESVD) of P and Q, respectively.(So V_i is $n \times n$, U_i is $m \times n$ and Σ_i are $n \times n$ and positive definite). Then from $PP^T = QQ^T$ we have:

$$V_1\Sigma_1^2 V_1^T = V_2\Sigma_2^2 V_2^T$$

Hence:

$$(\Sigma_2^{-1}V_2^T V_1\Sigma_1)(\Sigma_2^{-1}V_2^T V_1\Sigma_1)^T = I_{n\times n}$$

That is $(\Sigma_2^{-1}V_2^T V_1\Sigma_1) = Z_{n\times n}$ for some orthogonal matrix Z. So we have:

$$V_1\Sigma_1 U_1^T = V_2\Sigma_2 U_2^T(U_2 Z U_1^T)$$

$U = U_2 Z U_1^T$ is orthogonal so $P = QU$.$\triangle$

Theorem 2.2 *: Let $\vec{\mathbf{x}} = A_{m\times n}\vec{\mathbf{s}}$ with standard ICA assumptions and let $R_{\mathbf{xx}}$ be the correlation matrix of $\vec{\mathbf{x}}$. Then there exists a matrix $R_{\mathbf{xx}}^{-\frac{1}{2}}$ such that the vector $\vec{\mathbf{y}} = R_{\mathbf{xx}}^{-\frac{1}{2}}\vec{\mathbf{x}}$ is white (i.e. its correlation matrix is the identity) and with the additional assumption that $R_{\mathbf{ss}} = I_{n\times n}$ we have $\vec{\mathbf{y}} = U_{n\times n}\vec{\mathbf{s}}$ for some orthogonal matrix U.*

Proof: Let $R_{\mathbf{xx}} = V_{m\times n}\Lambda_{n\times n}V^T$ be the ESVD of $R_{\mathbf{xx}}$. Note that by the assumptions that A is full rank, $m \geq n$ and that all the components of $\vec{\mathbf{s}}$ have nonzero variance, $R_{\mathbf{xx}}$ has rank n. Let's define $R_{\mathbf{xx}}^{-\frac{1}{2}} = \Lambda^{-\frac{1}{2}}V^T$. Obviously $R_{\mathbf{yy}} = R_{\mathbf{xx}}^{-\frac{1}{2}}R_{\mathbf{xx}}R_{\mathbf{xx}}^{-\frac{T}{2}} = I_{n\times n}$.
Now we have

$$R_{\mathbf{xx}} = AR_{\mathbf{ss}}A^T = AR_{\mathbf{ss}}^{\frac{1}{2}}(AR_{\mathbf{ss}}^{\frac{1}{2}})^T = V\Lambda^{\frac{1}{2}}(V\Lambda^{\frac{1}{2}})^T$$

So using the previous lemma $AR_{\mathbf{ss}}^{\frac{1}{2}} = V\Lambda^{\frac{1}{2}}U_{n\times n}$ for some orthogonal U. Then we can write $\Lambda^{-\frac{1}{2}}V^TAR_{\mathbf{ss}}^{\frac{1}{2}} = R_{\mathbf{xx}}^{-\frac{1}{2}}AR_{\mathbf{ss}}^{\frac{1}{2}} = U$ or $R_{xx}^{-\frac{1}{2}}A = UR_{ss}^{-\frac{1}{2}}$. Thus

$$\vec{\mathbf{y}} = R_{\mathbf{xx}}^{-\frac{1}{2}}\vec{\mathbf{x}} = R_{\mathbf{xx}}^{-\frac{1}{2}}A\vec{\mathbf{s}} = UR_{\mathbf{ss}}^{-\frac{1}{2}}\vec{\mathbf{s}}$$

But, $R_{ss}^{-\frac{1}{2}}$ is diagonal, and due to the scale indeterminacy in the ICA problem we can assume that it is part of the source scaling. So we may write, $\vec{\mathbf{y}} = U\vec{\mathbf{s}}$.$\triangle$
Note that the assumption $R_{\mathbf{ss}} = I_{n\times n}$ is not restrictive due to scale ambiguity in the ICA problem. As a matter of fact it is more a convention than an assumption. Reviewing the above proof reveals that we can give an immediate generalization of this theorem to "*sphering in cumulant slices*", as follows:

Theorem 2.3 *(Generalized sphering): Let $\vec{\mathbf{x}} = A_{m\times n}\vec{\mathbf{s}}$ with standard ICA assumptions and let C be any cumulant slice matrix that is positive semi-definite.*

Then there exists a matrix $C^{-\frac{1}{2}}$ *such that for the vector* $\vec{\mathbf{y}} = C^{-\frac{1}{2}}\vec{\mathbf{x}}$ *we can assume* $\vec{\mathbf{y}} = U_{n\times n}\vec{\mathbf{s}}$ *for some orthogonal matrix* U*.*

Proof: Proof is essentially the same as the previous one.△

2.4.4 Whitening and Independence

An interesting question is: do we reduce the mutual information by decorellating the data?. The following example shows that it is not the case in general:

Example1: Let's consider two sources $\mathbf{x}_1$ and $\mathbf{x}_2$ each uniformly distributed in$(-\frac{1}{2}, \frac{1}{2})$. Because the mutual information is invariant with respect to multiplication by diagonal and permutation matrices we consider two different types of mixing matrices: one of the form $A_1(\theta) = \begin{bmatrix} \cot(\theta) & 1 \\ 1 & \cot(\theta) \end{bmatrix}$ and the other of the form $A_2(\theta) = \begin{bmatrix} \cot(\theta) & -1 \\ 1 & \cot(\theta) \end{bmatrix}$ where $\theta \in (0, \frac{\pi}{4})$. Obviously A_2 is equivalent(as far as the mutual information is concerned) to a rotation by θ. We consider mixing the sources under both of these matrices and evaluate the mutual information of the mixture. Figure (2.1) shows the mutual information in terms of θ under the two mixers. The graph shows that mixing under A_2 has a supremum mutual information of about .3 for $\theta = \frac{\pi}{4}$. On the other hand there is a θ_c below which the mixture under A_1 has mutual information less than 0.3. So if the sources are mixed with A_1 with $\theta < \theta_c$ which corresponds to the case that "the mixing matrix is more diagonal than certain amount" then by whitening the data we *may* increase the mutual information. Certainly there is a whitening matrix that makes the data independent but there is also a whitening matrix that increases the mutual information in this case. Beyond θ_c all the whitening matrices reduce the mutual

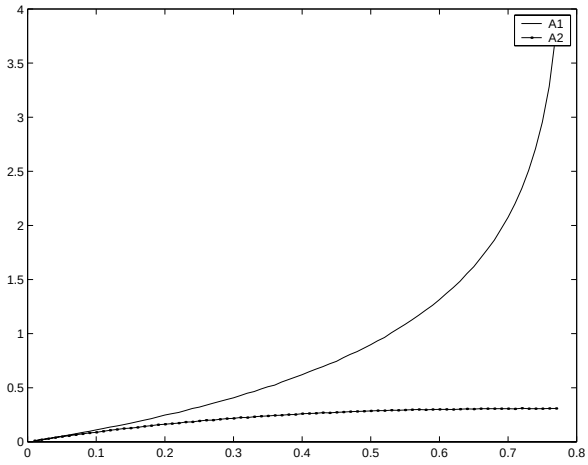

Figure 2.1: Mutual information of mixtures $A_i(\theta)\vec{\mathbf{x}}$ in terms of θ, in Example 1

information. It should be noted that the interesting fact that there is a maximum achievable mutual information using orthogonal mixing matrix is an immediate consequence of the compactness of the group of orthogonal matrices.$\triangle$

The conclusion of the example is that in the case where the mixture is such that the sources have almost equal power contribution in the received signals then by whitening the data, the mutual information is reduced. In this case and if the number of sources is large, we can argue that the received signal $\vec{\mathbf{x}}$, due to Central Limit Theorem [Papoulis] is almost Gaussian. Thus, the second term in (2.5) is already zero and by making the last term equal to zero $I[B\vec{\mathbf{x}}]$ will reduce. On the other hand if one source is dominant in each sensed signal, i.e. the mixture is not fully mixed, by whitening we may mix more and increase the mutual information. Thus in the generic case, i.e. in the case that signals are mixed enough and have almost equal power contribution in the received signals, whitening or PCA can be considered as a first step in the ICA in order to reduce dependence. Moreover

sphering has other benefits that we will consider in Section 7.1.

2.4.5 A Contrast for White Signals

Using (2.4) we can have an approximation of the mutual information in terms of cumulants. For the whitened signal $\vec{\mathbf{y}} = U\vec{\mathbf{s}}$ where U is orthogonal and $R_{ss} = I_{n\times n}$, by (2.4) and (2.5) we can have this approximation of the mutual information in terms of the cumulants:

$$I[\vec{\mathbf{y}}] \simeq N[U\vec{\mathbf{s}}] - \frac{1}{48}\sum_{i=1}^{n}(4\kappa_{iii}^2 + \kappa_{iiii}^2 + 7\kappa_{iii}^4 - 6\kappa_{iii}^2\kappa_{iii})$$

where $\kappa_{iii} = Cum_{\mathbf{y}}(i,i,i)$ and $\kappa_{iiii} = Cum_{\mathbf{y}}(i,i,i,i)$. Each of these cumulants using C1 can be written as $\kappa_{iii} = \sum_{j=1}^{n} u_{ij}^3 Cum_s(i,i,i)$ and $\kappa_{iiii} = \sum_{j=1}^{n} u_{ij}^4 Cum_s(i,i,i,i)$ where u_{ij} is the ij^{th} element of the orthogonal matrix U. Afterwards, considering the fact that $N[U\vec{\mathbf{s}}]$ is independent of U minimizing the mutual information is equal to maximization of $\Psi(U) = \sum_{i=1}^{n}(4\kappa_{iii}^2 + \kappa_{iiii}^2 + 7\kappa_{iii}^4 - 6\kappa_{iii}^2\kappa_{iii})$ with respect to the elements of the orthogonal matrix U. Whether $\Psi(U)$ is such that its minimization results in independence is not known [Comon 1], however a simplification of $\Psi(U)$ is proved to be a contrast [De Lathauwer2]:

Theorem 2.4 *: Let $\vec{\mathbf{x}} = U_{n\times n}\vec{\mathbf{s}}$, and let U be doubly stochastic, i.e. the 2-norm of each row or column of U is one and also assume that $\vec{\mathbf{s}}$ has finite cumulants up to order r and has at most one zero cumulant of order r. Let $\kappa_{r,i}$ be the r^{th} order cumulant of x_i, and $\Upsilon_{1,r}(U) = \sum_{i=1}^{n} |\kappa_{r,i}|$ and $\Upsilon_{2,r}(U) = \sum_{i=1}^{n} \kappa_{r,i}^2$. If either of $\Upsilon_{1,r}(U)$ or $\Upsilon_{2,r}(U)$ is maximized then U is a permutation, that is $\Upsilon_{1,r}$ and $\Upsilon_{2,r}$ are contrast functions over the set of doubly stochastic matrices for random vectors satisfying above conditions.*

Again we note that the set of doubly stochastic matrices is a compact set and hence the cost function defined has global minima on it. Because of the fact that any orthogonal matrix is also a doubly stochastic matrix we have this corollary:
Corollary: With the above conditions, if U is orthogonal then $\Upsilon_{1,r}$ and $\Upsilon_{2,r}$ are contrast functions.

Note that without some sort of compactification, optimizing functions of cumulants is meaningless, because cumulants are not scale invariant. For this reason, most of the contrast function introduced in the literature are over the set of orthogonal matrices (see for example [Cardoso1], [Comon1], [Moreau]).
It is interesting to note that maximizing $\Upsilon_{2,r}(U)$ for orthogonal U is equivalent to minimizing the sum of the square of the cross cumulants, which is sensible because cross cumulants measure dependence among variables.
There are other contrast functions, among them we will consider the JADE [Cardoso1] contrast function, in the next chapter.

2.5 The Group Structure of the ICA Problem

An important property of the standard ICA problem in the case $m = n$ is its group or multiplicative structure [Haykin], [Cardoso 2]. By the group structure we mean that starting from $\vec{\mathbf{x}} = A\vec{\mathbf{s}}$, A non-singular, if for a nonsingular matrix K we form $\vec{\mathbf{y}} = K\vec{\mathbf{x}} = KA\vec{\mathbf{s}} = C\vec{\mathbf{s}}$ then the form of the problem has not changed and we have not missed or gained any information. Obviously if A and K are orthogonal then C also is orthogonal. In fact many algorithms use this fact. For example in [Cardoso1] and many other papers, an iterative optimization method is used that updates the orthogonal un-mixing matrix U_k as $U_{k+1} = H_k U_k$ where H_k is a proper Jacobi rotation matrix. So this way we flow over the group of orthogonal matrices

at each step as the cost function is reduced. In Chapter 4 we will introduce the concept of gradient flow over groups, which is useful in contrast optimization.

The group structure has another implication that is called *equivariance* or *uniform performance* property, which enables us to devise estimators whose finite sample performance in the noiseless case is independent of the mixing matrix, that is they will give the same answer for different mixing matrices. Specifically, let X_T be a matrix of T realizations of $\vec{\mathbf{s}}$ then the estimator $\mathcal{A}(S_T)$ is equivariance if $\mathcal{A}(AS_T) = A\mathcal{A}(S_T)$. Then for the ICA problem $\vec{\mathbf{x}} = A\vec{\mathbf{s}}$, the estimated sources will be:

$$\hat{S}_T = \mathcal{A}^{-1}(AS_T)\, AS_T = \mathcal{A}^{-1}(S_T)A^{-1}AS_T = \mathcal{A}^{-1}(S_T)S_T$$

which depends only on the sources and not the mixing matrix. The key point here is that if we have contrast functions that depend only on the output vector $\vec{\mathbf{y}} = B\vec{\mathbf{x}}$ then this property is achieved. Because for two mixing matrices A_1 and A_2 and mixed signals $\vec{\mathbf{x}_1} = A_1\vec{\mathbf{s}}$ and $\vec{\mathbf{x}_2} = A_2\vec{\mathbf{s}}$ if B_1 and B_2 minimize a function of $\vec{\mathbf{y}_1}$ and $\vec{\mathbf{y}_2}$, respectively; then $B_1A_1 = B_2A_2$, that is the restored signal will be the same. Evidently, this is due to the group (multiplicative) structure of the problem.

2.6 Measures of Performance

The performance of different BSS/ICA algorithms can be compared based on their separation ability, the computational cost or the conditions under which they perform well. However in this section by "Performance Measure" we mean separation performance, that is how well the sources are separated by an algorithm. In practice there are different performance measures introduced (see Chapter 5 in [Haykin]). Some of them try to measure the interference and noise at the output of the separator whereas some others try to measure how far the estimated un-

mixing matrix is from a true one. In the case of noisy ICA these two methods are not equivalent, because asymptotically we might be able to estimate a true un-mixing matrix (using only HOS) but by applying that un-mixing matrix the output noise is not cancelled, that is the interference can be nulled, but the noise can not. In this thesis we follow the second path, because it is straightforward to compute therefore will compare algorithms on their performance in finding the un-mixing matrix in noisy ICA. In fact with the usual assumptions in the standard ICA model we can not do so much about noise other than cancelling its effect on estimating the un-mixing matrix.

Let B be the estimated un-mixing matrix and let $P = BA$. If $P = \Pi\Lambda$ where Λ is a non-singular diagonal matrix and Π is a permutation matrix, then the un-mixing matrix is estimated perfectly. We can measure the distance of P from permuted diagonal matrices, for example as [Hyvarinen]:

$$Index(P) = \sum_{i=1}^{n}(\sum_{j=1}^{n}\frac{|p_{ij}|}{\max_k |p_{ik}|} - 1) + \sum_{j=1}^{n}(\sum_{i=1}^{n}\frac{|p_{ij}|}{\max_k |p_{kj}|} - 1) \qquad (2.6)$$

Obviously the smaller $Index(P)$ the better the performance is and in fact it is zero if and only if P is a permuted diagonal matrix. In performing simulations we can average $Index(P)$ for many different realizations and different sources to get an over all performance assessment of an ICA algorithm.

2.7 A Survey of ICA Algorithms

There are many different classes of algorithms and schools of thought in solving ICA/BSS problem. One categorization is to divide algorithms in two classes: *online* and *off-line* or *batch* algorithms. Online algorithms refer to the class of algorithms that process the data on a sample-by-sample basis, whereas batch or off-line meth-

ods refer to those algorithms that process blocks of data. In the ICA context online algorithms usually implement a *stochastic gradient* or another form of stochastic optimization methods. The Amari's Natural Gradient (see Chapter 2 in [Haykin]) or its equivalent Cardoso's Relative Gradient [Cardoso2] is an online algorithm that uses the stochastic gradient over group of nonsingular matrices, i.e. the iterative algorithm updates the unknown matrix by multiplicative updates. FASTICA is an online algorithm that uses a Newton optimization method [Hyavarinen]. The category of off-line methods are usually cumulant based, whereas the online algorithms in addition to the cumulants use some other nonlinear functions of the data. Comon's ICA [Comon1] and Cardoso's JADE [Cardoso1] are among the most well known batch algorithms. There are also algorithms based on Joint Diagonalization of cumulant slices and tensor. In [Yeroder] and [Ziehe] ICA by means of joint diagonalization of cumulant slices by non-orthogonal matrices is addressed and in this thesis we give alternatives for their methods. In [De Lathauwer1], methods based on the idea of tensor diagonalization for cumulant tensors are developed .

On the other hand there are algorithms that deal with extensions of the basic ICA/BSS model. For instance Pham and Cardoso and developed algorithms based on joint diagonalization of a set of correlation matrices for BSS/ICA of non-stationary sensors [Pham1]. There are methods to deal with the case of more sensors than sources (see Chapter 16 in [Hyavarinen]) . There are schemes to consider sources with discrete values, i.e. sources that have values in a finite set, as in [Grellier].

Chapter 3

The Joint Diagonalization Criterion

In this chapter we introduce the Joint Diagonalization (JD) of cumulant slices criterion and its application in the ICA/BSS problem. We introduce the JADE algorithm which is a JD algorithm seeking orthogonal un-mixing matrix. We then extend the cost function of JADE to be able to search for non-orthogonal un-mixing matrices which is more desirable in the noisy ICA.

3.1 Joint Diagonalization of Cumulant Slices of White Signals

As it was mentioned in Chapter 2, in noiseless ICA it is possible to reduce the search space for the un-mixing matrix to the set of orthogonal matrices denoted by $\mathrm{O}(n)$, by whitening the data. The main advantage of this is that $\mathrm{O}(n)$ is a *compact multiplicative group* (in fact it is a compact Lie group) [Helmke]. Because

of its compactness we can define cost functions that achieve their optima on $\mathrm{O}(n)$. Consider $\vec{\mathbf{x}} = A_{m\times n}\vec{\mathbf{s}}$, with the assumptions $m \geq n$, A full rank and $\vec{\mathbf{s}}$ with independent components having at most one component with zero fourth order cumulant (i.e. assumptions S4, S5.3 from Section 2.1.1). After whitening $\vec{\mathbf{x}}$, by Theorem 2.2 we will have:

$$\vec{\mathbf{y}}(t) = U\vec{\mathbf{s}} \quad U_{n\times n} \in \mathrm{O}(n) \tag{3.1}$$

Using Lemma 2.1, for fourth order cumulant slices of the form $Cum_x(:,:,i,j)$ we have:

$$Cum_{\mathbf{y}}(:,:,i,j) = U\Lambda_{ij}U^T \tag{3.2}$$

where

$$\Lambda_{ij} = \begin{bmatrix} u_{i1}u_{j1}Cum_{\mathbf{s}}(1,...,1) & 0 & \cdots \\ 0 & u_{i2}u_{j2}Cum_{\mathbf{s}}(2,...,2) & \cdots \\ 0 & \ddots & 0 \\ \cdots & 0 & u_{in}u_{jn}Cum_{\mathbf{s}}(n,...,n) \end{bmatrix}$$

So remembering the fact that $Cum_{\mathbf{y}}(:,:,i,j)$ is a symmetric matrix, we can say that (3.2) is an *Eigen Decomposition* of the symmetric matrix $Cum_{\mathbf{y}}(:,:,i,j)$. On the other hand if the eigenvalues of $Cum_{\mathbf{y}}(:,:,i,j)$ are distinct we know that the diagonalizer U is unique up to a column permutation. We notice that with two conditions:

- U is a generic (random) orthogonal matrix
- At most one of the $Cum_{\mathbf{s}}(i,i,i,i)$ is zero (i.e. zero fourth order cumulant)

the eigenvalues of $Cum_{\mathbf{y}}(:,:,i,j)$ are distinct and hence any diagonalizer of it is essentially U. So by finding the eigen decomposition of any of these cumulant slices we can find U. However, in practice we estimate the cumulants by their sample averages, so we should not expect that two different cumulant slices have the same diagonalizer, especially because the estimation of cumulants is very non-robust. Hence, we may *jointly diagonalize* all or a subset of the cumulant matrix slices.

In [Cardoso1], a cost function for JD of a set of cumulant slices is introduced. Let $\{C_i\}_{i=1}^N$ be a subset of the set of 4^{th} order cumulant slices of $\vec{\mathbf{y}}$. Then the problem of finding a joint diagonalizer for $\{C_i\}_{i=1}^N$ can be considered as a minimization problem:

$$\min_{\Theta \in \mathrm{O(n)}} J_1(\Theta) = \sum_{i=1}^{N} \left\| \Theta C_i \Theta^T - \mathrm{diag}(\Theta C_i \Theta^T) \right\|_F^2 \tag{3.3}$$

where $\mathrm{diag}(X)$ is a diagonal matrix whose diagonal is identical to X's diagonal and $\|.\|_F$ is the matrix Frobenius norm operator. In fact this is a constrained optimization problem over the *group* $\mathrm{O}(n)$. In different occasions we will see how the group structure of the constraint set can be exploited.

Remark: As it is clear, in general there exists no exact diagonalizer, so a more accurate title for the subject is "*Joint Approximate Diagonalization* (JAD)" as it was and still is used in the literature. However by "Joint Diagonalization" we also imply the same concept so we will mainly use the latter.

3.2 The JADE Algorithm

The Joint Approximate Diagonalization of Eigen matrices or JADE is one of the earliest ICA algorithms to use the idea of joint diagonalization of cumulants slices. The algorithm can be divided in two separate parts: first, finding a set of symmetric matrices to be diagonalized and second a joint diagonalization algorithm based on

Jacobi rotations. We will first describe the joint diagonalization algorithm, because it turns out that this is the most applicable and versatile part of the algorithm. It should be noted that in the literature this joint diagonalization algorithm, itself is called the JADE algorithm, so with some abuse of terminology, later on we will refer to this algorithm as JADE.

3.2.1 Orthogonal Joint Diagonalization Jacobi Rotations (the so-called JADE Algorithm)

The method of Jacobi or Givens rotations in the context of finding the eigen values of a symmetric matrix is a well established method [Golub]. An extension of this method can be used to find an approximate orthogonal joint diagonalizer for a set $\{C_i\}_{i=1}^N$ of symmetric matrices. The idea is to compute Θ by *orthogonal multiplicative* updates such that at each step the cost function J_1 is reduced. The orthogonal multiplicative updates keep the iterate always orthogonal. These are matrices known as Givens or Jacobi rotations which have this simple form:

$$\Theta_{kl} = \begin{bmatrix} 1 & 0 & \dots & \downarrow^{k} & \dots & \downarrow^{l} & \dots \\ 0 & 1 & 0 & 0 & \dots & 0 & \dots \\ & \dots & \ddots & \vdots & & \vdots & \\ {}^{k}\longrightarrow & 0 & \cdots & \cos\theta_{kl} & \cdots & -\sin\theta_{kl} & \cdots \\ & & & \vdots & \ddots & \vdots & \\ {}^{l}\longrightarrow & 0 & \cdots & \sin\theta_{kl} & \cdots & \cos\theta_{kl} & \cdots \\ & & & \vdots & & \ddots & \\ & \dots & & 0 & \dots & 0 & 1 \end{bmatrix} \tag{3.4}$$

where θ_{lk} is an angle that at each step is computed in order to minimize the cost function. In [Cardoso1] the algorithm is proposed for complex and not necessarily

symmetric matrices, here we give a version for symmetric and real matrices. The algorithm is coded in Table (3.1).

Algorithm 3.1:

1. Consider the set of $n \times n$ symmetric matrices $\{C_i\}_{i=1}^N$, which for notational convenience we denote as $\{C^i\}_{i=1}^N$, let $\Theta = I_{n\times n}$.

2. For $1 \leq k < l \leq n$ do:

Form the $N \times 2$ matrix:

$$G_{kl} = \begin{bmatrix} c_{kk}^1 - c_{ll}^1 & 2c_{kl}^1 \\ c_{kk}^2 - c_{ll}^2 & 2c_{kl}^2 \\ \vdots & \vdots \\ c_{kk}^N - c_{ll}^N & 2c_{kl}^N \end{bmatrix}$$

Compute $\vec{v}_{2\times 1}$ a unit-norm eigen vector of $G_{kl}^T G_{kl}$ corresponding to the larger eigen value and find θ_{kl} from $\vec{v} = [\cos 2\theta_{kl} \quad \sin 2\theta_{kl}]^T$

Form Θ_{kl} from (3.4) and update $\Theta \longleftarrow \Theta_{kl}\Theta$ and $C^i \longleftarrow \Theta_{kl} C^i \Theta_{kl}^T$.

3. **If** "required" **goto** step 2 **else** stop.

Table 3.1: The so-called JADE algorithm for joint diagonalization of the set $\{C^i\}_{i=1}^N$ by an orthogonal matrix. The unspecified parameters and qualities are to be decided in practice.

Each run of the step 2 is called a *sweep*. The appearance of an eigen vector of G_{kl} is a result of a simple constrained quadratic minimization of $J_1(\Theta_{kl})$ which is a function of only θ_{kl} . We note that in the light of the group structure of the problem, a sequential optimization is possible and the orthogonality of Θ is guaranteed by construction.

3.2.2 The JADE Algorithm

Now we state the original JADE algorithm for the BSS problem. The steps in the original JADE algorithm are as follows:

1. Whiten the mixed signal $\vec{\mathbf{x}}$ to get $\vec{\mathbf{y}} = R_{\mathbf{xx}}^{-\frac{1}{2}}\vec{\mathbf{x}}$.

2. Estimate $Cum_{\mathbf{y}}$ the 4^{th} order cumulant tensor of $\vec{\mathbf{y}}$.

3. Find $\{E_i\}_{i=1}^{n}$, the eigen matrices corresponding to the n largest eigen values of Cum_y.

   Remark: Note that any 4^{th} order n-dimensional tensor is a linear mapping from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{n\times n}$, i.e. from the space of $n \times n$ matrices to the space of $n \times n$ matrices. If this mapping is symmetric or self-adjoint, as a cumulant tensor is , then there exist n^2 eigen matrices that are orthogonal to each other and can represent the tensor. With the condition $\vec{\mathbf{y}} = U\vec{\mathbf{s}}$ and the independence of the components of $\vec{\mathbf{s}}$, it is shown in [Cardoso1] that only n eigen values of $Cum_{\mathbf{y}}$ are nonzero, and also it is shown that the joint diagonalization of n^2 matrix slices of $Cum_{\mathbf{y}}$ is equivalent to the joint diagonalization of $\{E_i\}_{i=1}^{n}$.

4. Use Algorithm 3,1, to find Θ the orthogonal joint diagonalizer of $\{E_i\}_{i=1}^{n}$, by minimizing the cost function:

$$J_1(\Theta) = \sum_{i=1}^{n} \left\| \Theta E_i \Theta^T - \mathrm{diag}(\Theta E_i \Theta^T) \right\|_F^2$$

In practice because the linear model may not hold exactly and that the cumulant estimates are not accurate, step 3 is not justified and is neglected and that is why we did not delve more into computing eigen matrices. Hence in step 4 a set of matrix slices of $Cum_{\mathbf{y}}$ is used, for example the set $\mathcal{N} = \{Cum_{\mathbf{y}}(:,:,i,j) | 1 \leq i,j \leq n\}$.

Remark: It is possible to associate weights to each term in the above cost function. Hence a more general form of J_1 is:

$$J_1(\Theta) = \sum_{i=1}^{n} w_i \left\| \Theta C_i \Theta^T - \text{diag}(\Theta C_i \Theta^T) \right\|_F^2$$

where $w_i > 0$ and $\sum_{i=1}^{N} w_i = 1$. Note that this is equivalent to considering the set $\{\sqrt{w_i} C_i\}_{i=1}^{N}$ instead of $\{C_{i=1}\}_{i=1}^{N}$ in the original form of J_1. Hence the cost function with weights can be reduced to one without weights.

3.3 Non-Orthogonal Joint Diagonalization

The main drawback of the JADE algorithm and other ICA algorithms that whiten the data and confine the search space to orthogonal matrices is that in the presence of noise the whitening process is biased, due to the fact the noise covariance is unknown, and this bias cannot be compensated in the subsequent orthogonal search. On the other hand the 4^{th} order cumulants are blind to Gaussian noise, so a method that uses only higher order cumulants can suffer less from this problem. One approach based on JD is to search for a non-orthogonal joint diagonalizer. This can be justified by Lemma 2.1 which states that all the cumulant matrix slices of any order are diagonalizable in a congruence manner by the pseudo-inverse of the mixing matrix. To be able to have an algorithm for joint diagonalization of a set of matrices we should have suitable cost functions for joint diagonalization.
In developing JD cost functions for a set of symmetric matrices $\{C_i\}_{i=1}^{N}$, we may follow two approaches:

1. Find a matrix B such that $\{BC_iB^T\}_{i=1}^{N}$ are as diagonal as possible, or
2. Find W and diagonal matrices $\{\Lambda_i\}_{i=1}^{N}$ such that $C_i \simeq W\Lambda_i W^T$ as much as possible for all $1 \leq i \leq N$.

The first approach deals with the un-mixing matrix directly but the second one looks for an estimate of the mixing matrix. The second approach has the drawback that because if $C_i \simeq W\Lambda_i W$ then $C_i \simeq (W\Lambda^{-1})\,\Lambda\Lambda_i\Lambda\,(W\Lambda^{-1})^T$, for any diagonal matrix Λ, it is possible that W is very bad conditioned, then its inverse will be hard to find.

We can use different measures for approximating the "closeness". One such measure can be the Frobenius norm of the error, which has the advantage of analytical convenience. So we can have the corresponding cost functions for the first approach as:

$$\min J_1(B) = \sum_{i=1}^{N} \left\| BC_iB^T - \mathrm{diag}(BC_iB^T) \right\|_F^2 \tag{3.5}$$

where B is a full row-rank $n \times m$ matrix (we assume $n \leq m$). Note that this is exactly the cost function we used for the orthogonal case, but now on a larger space. A cost function corresponding to the second approach used in [Yeredor] is

$$\min J_2(W, \{\Lambda_i\}_{i=1}^N) = \sum_{i=1}^{N} \left\| C_i - W\Lambda_i W^T \right\|_F^2 \tag{3.6}$$

where W belongs to the manifold of full column-rank $m \times n$ matrices and Λ_i are diagonal $n \times n$ matrices.

3.3.1 Square Mixing Matrices

So far we have considered rectangular mixing matrices, from now on we consider only square mixing matrices. In fact this restriction is not very prohibitive, because by whitening the data we can reduce the number of sensors to the number of sources. On the other hand we may argue that we could use the redundancy in the data in cumulant domain as well. The main reason that we consider only square matrices is that in the subsequent chapters we intend to use the group

structure of the BSS/ICA problem and develop methods in matrix groups, hence the assumption of having an inverse in the group-theoretic sense is important and only square matrices have this property. *So from now on we assume that the mixing matrix is square ($m = n$) unless otherwise stated.*. Yet it is possible to extend the idea of group structure in a similar form to non-square case as in [Zhang].

3.3.2 Two Desired Properties for JD Cost Functions

In the context of ICA/BSS we expect a cost function for JD to have two properties:

1. It should be scale invariant, i.e. it should not change by diagonal mixing or un-mixing matrices,

2. It should be invariant under permutations, i.e. it should not change by permutation mixing or un-mixing matrices.

These two properties represent the inherent indeterminacies in the ICA/BSS problem, and resemble the properties of the mutual information. We recall that $I[\vec{\mathbf{x}}] = I[\Lambda\vec{\mathbf{x}}]$ for any non-singular diagonal matrix Λ. Obviously J_1 does not have the first property but has the second one. J_2 on the other hand has the first property in an extended sense, because $J_2(W\Lambda, \{\Lambda^{-1}\Lambda_i\Lambda^{-1}\}_{i=1}^N) = J_2(W, \{\Lambda_i\}_{i=1}^N)$ for any non-singular $W_{n\times n}$, i.e. we can keep the cost function unchanged under diagonal mixing. J_2 is obviously unchanged under permutations as well.

Remark: We can derive JD cost functions either in terms of an un-mixing matrix B or the (estimated) mixing matrix W. If J is a scale-invariant JD cost function then $J(\Lambda) = J(I_{n\times n})$. If J is in terms of B this translates to $J(\Lambda B) = J(B)$ and if it is in terms of W it translates to $J(W\Lambda) = J(W)$. Therefore scale-invariance in terms of the un-mixing matrix refers to multiplication by a diagonal matrix from

the left and in terms of the mixing matrix it refers to multiplication from the right by a diagonal matrix. In this thesis we are mainly interested in cost functions in terms of un-mixing matrices.

3.3.3 Examples of Scale and Permutation Invariant Cost Functions

In this section we introduce some cost functions for non-orthogonal JD that have the scale and permutation invariance property.

Example 1: $J_1(B)$ is not scale invariant on the set of $n \times n$ non-singular matrices as mentioned above, but it is scale invariant if we restrict B to belong to $\mathrm{O}(n)$. It is permutation invariant on the set of $n \times n$ non-singular matrices

Example 2 [Pham2]: The cost function

$$J_3(B) = \sum_{i=1}^{N} \log \big(\frac{\det \operatorname{diag}(BC_iB^T)}{\det BC_iB^T}\big)$$

is not well defined in general, but if $\{C_{i=1}\}_{i=1}^{N}$ are positive definite, then it is always non-negative and reaches zero when all $\{C_i\}$ is jointly diagonalizable. This cost function has the property that is scale invariant : and also $J_3(\Lambda B) = J_3(B)$, for any non-singular diagonal Λ. It scale-invariant too.

Example 3: Another form of the above cost function with Frobenius norm is

$$J_4(B) = \sum_{i=1}^{N} \big\| BC_iB^T(\operatorname{diag}(BC_iB^T))^{-1} - I \big\|_F^2$$

This cost function is scale invariant if C_i's are positive definite. Note that this is required to ensure that $\operatorname{diag}(BH_iB^T)$ is invertible. We also have $J_4(\Lambda B) = J_4(B)$ for any non-singular diagonal Λ. It is also permutation invariant.

Example 4: A cost function that somehow normalizes J_1 is

$$J_5(B) = \sum_{i=1}^{N} \left\| C_i - B^{-1}\text{diag}(BC_iB^T)B^{-T} \right\|_F^2 \tag{3.7}$$

This cost function does not require positive definiteness of C_i. It is scale and permutation invariant. We also have $J_5(\Lambda B) = J_5(B)$, for any non-singular diagonal Λ. Its drawback is that it uses B^{-1} together with B which imposes more computational cost in its minimization.

Remark: Some of the above cost functions require that C_i be positive definite. In fact as we mentioned before cumulant slices are not necessarily positive definite so these cost functions are not suitable for joint diagonalization of cumulant slices, but they can be used to jointly diagonalize a set of correlation matrices. Joint diagonalization of correlation matrices is a useful tool in separation of non-stationary sources [Pham1].

3.3.4 Dealing with Cost Functions that are not Scale-Invariant

The main problem with a joint diagonalization cost function that like J_1 is not scale invariant is that it can be reduced by a diagonal matrix, which in the context of BSS/ICA does not result in independence. In Chapter 6 we give some methods to deal with this issue. The main idea there is to somehow identify ΛB with B for all non-singular diagonal matrices Λ. This way we will exclude all un-mixing matrices that are essentially the same as B. This problem can be related also to the fact that the set of non-singular $n \times n$ matrices $\text{GL}(n)$ is not a compact set and a priori we have no guarantee that $J_1(B)$ has a minimum on $\text{GL}(n)$. In fact $J_1(B)$ has a global infimum of zero at $B = 0$ (which is not in $\text{GL}(n)$) and we will show in Section 6.1 that in the case that the set $\{C_i\}_{i=1}^N$ is not diagonalizable, J_1 does not have any local minima. The reason for this is simply the fact that at any

point the cost function can be reduced by a diagonal matrix. So the identification of B and ΛB can be considered as a method for compactification of $\mathrm{GL}(n)$. Another approach maybe to add penalty terms to the cost function which accounts for the non-compactness issue, for example:

$$\hat{J}_1(B) = J_1(B) - \log\left(|\det B|\right)$$

has the property that penalizes the cost when $|\det(B)|$ becomes small . This way we can exclude the global infimum of J_1 at $B = 0$.

Chapter 4

Matrix Lie Groups

In this chapter we briefly introduce the necessary tools from the theory of differentiable manifolds and Lie Groups needed in the subsequent chapters.

4.1 Introduction

The theory of *differentiable manifolds* is a part of the of field of *differential geometry*. Differential geometry is an important mathematical discipline in dealing with nonlinear systems. Lie theory deals with manifolds that have *group* structure. To define abstract groups we do not need any topological requirement, but *Lie groups* are in some sense continuous groups, i.e., close to any element of the group we can have another element. Exact treatment of differentiable manifolds and general Lie groups requires lots of technicalities and is quite complicated. In this chapter we consider only *matrix Lie* groups in order to furnish the necessary tools needed for our approach to the matrix joint diagonalization problem. We also give some results for differential equations on manifolds.

4.2 Riemannian Manifolds, Tangent Spaces and Gradients

For the purposes of this thesis we define a (smooth) manifold as follows:

Definition 4.1 *Consider the set $M^n \subset \mathbb{R}^N$. Assume that there exists a collection of open subsets of M^n such as $\{U_\alpha\}$ that cover M and each U_α is homeomorphic to an open set $\mathbb{R}^n$; i.e there exits a continuous and one-to-one map with continuous inverse, $\varphi_\alpha : U_\alpha \to \mathbb{R}^n$. Assume further that if $U_\alpha \cap U_\beta$ is non-empty then the transition maps $\varphi_\alpha \circ \varphi_\beta^{-1} : \varphi_\beta(U_\alpha \cap U_\beta) \to \varphi_\alpha(U_\alpha \cap U_\beta)$ and $\varphi_\beta \circ \varphi_\alpha^{-1} : \varphi_\alpha(U_\alpha \cap U_\beta) \to \varphi_\beta(U_\alpha \cap U_\beta)$ are smooth. The set M^n together will all possible such subsets and maps is called an n-dimensional smooth manifold. The set U_α is called a chart and φ_α is its corresponding (local) coordinate function and the function φ_α^{-1} is its corresponding (local) parameterization.*

Intuitively this means that M locally looks like $\mathbb{R}^n$, yet it is not flat in the same manner that $\mathbb{R}^n$ is. From this definition the graph of any smooth function $f : \mathbb{R}^n \to \mathbb{R}$ is an n dimensional smooth manifold. The same is true for a sphere in $\mathbb{R}^3$, which is a 2-dimensional manifold. Note that to describe the points on a sphere in the above form we need to use more than one local coordinate system to describe the whole sphere. On the other hand the graph of the function $|x|$, i.e. the points $(x, |x|)$ do not form a smooth manifold. There are manifolds more interesting than the ones mentioned. For example, the set of non-singular $n \times n$ matrices known as GL(n) is an n^2-dimensional manifold in $\mathbb{R}^{n^2}$, where singular matrices correspond to "holes" and are excluded from $\mathbb{R}^{n^2}$, or the set of orthogonal $n \times n$ matrices is an $\frac{n(n-1)}{2}$ dimensional manifold in $\mathbb{R}^{n^2}$. We will consider matrix manifolds in more detail later.

Remark: Note that from the above definition the collection of charts and coordinate functions are *maximal* in the sense that a manifold contains all the possible charts that are compatible with its structure. Therefore a point $p \in M$ we have in particular a pair (U_p, φ_p) with $\varphi_p(p) = 0 \in \mathbb{R}^n$.

4.2.1 Tangent Space

It should be noted that a manifold by its own does not need to have any algebraic properties and in fact manifolds are not in general closed under addition or multiplication or other algebraic operations. They are merely curved geometric objects that are smooth enough. Yet one can associate to them some sets that have nice properties. The most immediate one is a *tangent space* to a manifold at a point on the manifold. Tangent space, intuitively is nothing but the linear approximation of the manifold at a point, and it can be described based on the first order derivative of the local parameterization function:

Definition 4.2 *: Let $M^n \subset \mathbb{R}^{n+k}$ be a smooth manifold. Let φ be a local coordinate function on a neighborhood around p, then the columns of the Jacobian matrix $\frac{\partial \varphi^{-1}}{\partial \vec{x}}$ centered at p span a linear space of dimension n, which is called the tangent space at p denoted by T_pM and any vector in this space is called a tangent vector. The set $TM = \bigcup_{p \in M} T_pM$ is called the tangent bundle of M. A function $X : M \to TM$ that to any point $p \in M$ assigns tangent vector $X(p) \in T_pM$ in a smooth fashion is called a smooth vector field.*

It can be shown that the above definition is a geometric one and independent of the local coordinate function used. Tangent vectors are identified by their effect, which is to give the directional derivative of a function on the manifold in their

directions. It is possible to relate the above definition to the concept of a curve on a manifold. A curve is a one dimensional object defined as:

Definition 4.3 *: A curve through a point $p \in M$ is a smooth map $\gamma : (-b, b) \rightarrow M$, $b > 0$ such that $\gamma(0) = p$. Usually we denote the independent real variable in the definition as time t and the curve will be $\gamma(t)$.*

Let U_p be a neighborhood around p with the coordinate function φ. Let φ_i^{-1} denote the local parameterization function where all but i^{th} component are fixed to zero and x_i is the only variable. Then φ_i^{-1} defines a curve on M. Assume that $\gamma(t) \in U_p$ for $-b < t < b$ then the composition $\varphi \circ \gamma : (-b, b) \subset \mathbb{R} \rightarrow \varphi(U_p) \subset \mathbb{R}^n$ is a vector valued function of the variable t on an open interval around $t = 0$, so we can define its time derivative at t=0 in terms of the time derivative of each of its components $\varphi_i \circ \gamma$. The time derivative or velocity of the curve $\gamma(t)$ at $t = 0$ or at p, can be defined as

$$\left.\frac{d\gamma}{dt}\right|_{t=0} = \left.\frac{d(\varphi^{-1} \circ \varphi \circ \gamma)}{dt}\right|_{t=0} = \sum_{i=1}^{n} \left.\frac{\partial \varphi^{-1}}{\partial x_i}\right|_{x=\varphi(\gamma(0))} \left.\frac{d(\varphi_i \circ \gamma)}{dt}\right|_{t=0} =$$

$$\sum_{i=1}^{n} \left.\frac{d\varphi_i^{-1}}{dx_i}\right|_{x=\varphi(\gamma(0))} \left.\frac{d(\varphi_i \circ \gamma)}{dt}\right|_{t=0}$$

Thus the velocity vector of the curve $\gamma(t)$ at $t = 0$ is a linear combination of the the tangent vectors or the velocity vectors of the curves φ_i^{-1} at $t = 0$. Therefore it is a tangent vector as well. This definition is independent of the local coordinate used. Note that above is in fact an expansion of the velocity vector in the basis $\{\frac{\partial \varphi_i^{-1}}{\partial x_i}\}_{i=1}^n$, so the vector $[\frac{d(\varphi_1 \circ \gamma)}{dt}(0), ..., \frac{d(\varphi_n \circ \gamma)}{dt}(0)]^T$ is the representation of the velocity vector of γ at p corresponding to the local coordinate function φ. We can also define the directional derivative of a function $f : M^n \rightarrow \mathbb{R}$ along a curve γ at $\gamma(0) = p$ based on the *chain rule* as :

$$Df|_{\dot{\gamma}(0)} = (\gamma_* f)\ (p) = \left.\frac{d(f \circ \gamma)}{dt}\right|_{t=0} = \sum_{i=1}^{n} \left.\frac{\partial f(\varphi_i^{-1})}{\partial x_i}\right|_{x=\varphi(p)=0} \left.\frac{d(\varphi_i \circ \gamma)}{dt}\right|_{t=0}$$

Note that the directional derivative depends on how f changes on M at p and the velocity of the curve at p.

4.2.2 Riemannian Manifolds

We emphasize again that T_pM is a linear vector space at any point. Therefore we can equip T_pM at any point with an inner product:

Definition 4.4 *A Riemannian metric on M^n is a family of inner products $\langle ., . \rangle_p$ defined on each tangent space T_pM, such that it depends smoothly on $p \in M$. When endowed with a Riemannian metric, M is called a Riemannian manifold.*

We recall that any inner product on T_pM has a positive definite matrix representation. That is if two tangent vectors in $\xi, \eta \in T_pM$ have representations $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$, respectively, then $\langle \xi, \eta \rangle_p = \vec{u}^T Q_p \vec{v}$ where Q_p is the positive definite matrix representing the Riemannian metric at p.

Let f be a smooth function $f : M \to \mathbb{R}$. It can be shown that there exists a unique smooth vector field $\nabla f : M \to TM$, $p \mapsto \nabla f(p)$ such that for a point p and any smooth curve $\gamma(t)$ with $\gamma(0) = p$ we have $(\gamma_* f)\ (p) = \langle \nabla f(p), \dot{\gamma}(0) \rangle_p$ where $\dot{\gamma}(0) = \frac{d\gamma}{dt}(t = 0)$. This vector field is called the *gradient* vector field. Note that this vector field depends on the Riemannian metric used, and in fact the above equation is the defining equation for the gradient vector field. The result of this definition is that if the function f has a local minimum at $p_0 \in M$ then $\nabla f(p_0) = 0$ at that point. A point $p_0 \in M$ is called a critical point of f if $\nabla f(p_0) = 0$. On the

other hand we can define the *Hessian* $H_f(p_0)$ at a critical point p_0 as a symmetric bilinear form such that for any curve $\gamma(t)$ on M with $\gamma(0) = p$ we have:

$$H_f(p_0) : T_{p_0}M \times T_{p_0}M \to \mathbb{R}$$

$$H_f(p_0)(\dot{\gamma}(0), \dot{\gamma}(0)) = \frac{d^2}{dt^2}(f \circ \gamma)(0)$$

Note that due to the symmetry and multilinearity property of $H_f(p_0)$ we only need to specify it at points like $(\xi, \xi) \in T_{p_0}M \times T_{p_0}M$. A critical point p_0 for which the Hessian is positive definite, i.e. $H_f(p_0)(\dot{\gamma}(0), \dot{\gamma}(0)) > 0$ for any curve on the manifold passing through p_0 with nonzero velocity, is a *local minimum* of f on M. The significance of this formulation is that if in a constrained optimization problem the constraints form a smooth manifold then we can reformulate the optimality conditions without constraints and avoid Lagrange multipliers formulation. As we shall see later this enables us to consider complicated constraints such as orthogonality or non-singularity for matrices.

We mention that the Hessian is a bilinear form, which for an n-dimensional manifold can have a symmetric matrix representation. If for a critical point p_0 the corresponding Hessian matrix is non-singular then that point is called a *non-degenerate* critical point. Non-degenerate critical points are always isolated and there are at most countably many of them[Helmke].

4.3 Flows on Riemannian Manifolds

If a moving object has a velocity vector that is tangent to a manifold at any point then we expect that the object stays on the manifold. This is not true of course for all times because we can have the *finite escape time* phenomena, which means that the object may leave the manifold in finite time. For example if

$M = (0, 1)$ then the answer to the linear differential equation $\dot{x} = 1$ with $x(0) = \frac{1}{2}$ leaves M at $t = \frac{1}{2}$. An *integral curve* of the smooth vector field $X : M \to TM$ with initial condition $\gamma(0) = p$ is a differentiable map $\gamma : I_p \to M$ such that $\dot{\gamma_p}(t) = X(\gamma_p(t)), \forall t \in I_p$where I_p is an open interval containing 0. If the longest I_p is of finite length then we have the finite escape time phenomena.

The *flow* of X is the collection of all maps $\phi_t : M \to M$ such that $\phi_t(p) = \gamma(t)$ where $\gamma(t)$ is the integral curve with the initial condition $\gamma(0) = p$. Existence and uniqueness theorems guarantee that $\phi_t(p)$ is a diffeomorphism on-to its image with inverse $(\phi_t)^{-1} = \phi_{-t}$ [Helmke],[Marsden].

The ideas of stability for ordinary differential equations(ODEs) on manifolds are similar to those for ODEs on $\mathbb{R}^n$. Here we state a version of *La Salle's invariance principle* [Khalil] for flows on manifolds [Helmke]:

Theorem 4.1 ***La Salle's Invariance principle*** *Let $X : M \to TM$ be a smooth vector field on a Riemannian manifold and let $f : M \to \mathbb{R}$ be a smooth function such that it has compact sub-level sets, i.e. the set $\{p \in M | V(p) \leq c \ \forall c \in \mathbb{R}\}$ is compact and $\dot{f}(\gamma(t)) = \frac{d(f(\gamma(t)))}{dt} \leq 0$ for any solution $\gamma(t)$ of $\dot{\gamma}(t) = X(\gamma(t))(*)$. Then every solution of (*) stays on M for all $t \geq 0$. Moreover any solution approaches the largest compact, connected and invariant subset of $\Omega = \{p \in M | \langle \nabla f, X(p) \rangle_p = 0\}$.(f is called a weak Lyapunov function for (*)).*

An important class of flows on Riemannian manifolds is the class of *gradient flows*. If the smooth function $f : M \to \mathbb{R}$ has the gradient vector field (with respect to a certain Riemannian metric) ∇f then the differential equation $\dot{x} = -\nabla f(x)(**)$ is a flow in the negative direction of the gradient in order to minimize f. Obviously the critical points of f are equilibria of the flow. In fact we can see

that $f(x(t)) \leq f(x(0))$ for all $t \geq 0$ that $x(t)$ is defined on M. So f is a Lyapunov function for this flow. It can also been shown that if for a critical point the Hessian is positive definite then the point is asymptotically stable and if the Hessian has at least one negative eigenvalue then that point is unstable. Using the La Salle's theorem, we can show that if f has compact sub-level sets or M is compact and if f has only finite number of critical points then any solution of (**) converges to a single critical point of f (point-convergence as opposed to the set-convergence stipulated in the La Salle's theorem) and for typical initial conditions it converges to a local minimum. Nevertheless not all functions have compact sub-level sets or finite number of critical points. As we will see in the next chapters the convergence properties of this flow depend on both the manifold M and the function f.

4.4 Matrix Lie Groups

As it was mentioned before the set of non-singular $n \times n$ matrices $\mathrm{GL}(n)$ is a manifold. Actually in addition to being a manifold it is also a group under matrix multiplication. $\mathrm{GL}(n)$ is an example of a *Lie group*. A Lie group is a manifold G that has a group structure consistent with its manifold in the sense that the group multiplication is a smooth map. In this thesis we consider only matrix Lie groups, but it is true that "most, though not all, Lie groups can be realized as matrix groups" [HOWE]. There are different ways to treat Lie groups, here we choose a short and informal path (mainly borrowed from [HOWE]) and we give the results without proofs. This approach is based on defining the *exponential* map for matrices. For any $n \times n$ matrix A, $\exp A$ is defined as:

$$\exp A = e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

Note that this series converges for any matrix. Then the Lie algebra of matrix Lie group G is defined as: $\mathfrak{g} = \{\Delta \in \mathbb{R}^{n\times n} | \exp(t\Delta) \in G \text{ for all } t \in \mathbb{R}\}$. It can be shown that $\mathfrak{g}$ is a vector space. Moreover $\mathfrak{g}$ is closed under the operation $[.,.]$ defined on $\mathfrak{g} \times \mathfrak{g}$ as:

$$(\Delta_1, \Delta_2) \mapsto [\Delta_1, \Delta_2] = \Delta_1\Delta_2 - \Delta_2\Delta_1$$

$[.,.]$ is called the (matrix) *Lie bracket*. Interestingly, $\mathfrak{g}$ has a significant geometrical meaning: it is nothing but the tangent space to G at the identity matrix $I_{n\times n}$. On the other hand the tangent space at any other point $B \in G$ can be constructed by matrix multiplication from the elements of $\mathfrak{g}$ as: $T_BG = \{B\Delta | \Delta \in \mathfrak{g}\} = \{\Delta B | \Delta \in \mathfrak{g}\}$. Note that although a tangent vector can be produced by both left and right multiplication by B yet two different elements in $\mathfrak{g}$ should be used to produce the same tangent vector. In subsequent chapters we will use the structure of T_BG to define suitable Riemannian metrics that match the group structure.

As for any other manifold, we can define flows on matrix Lie groups. For example one class of flows over matrix manifolds is the exponential flow which corresponds to the differential equations $\dot{B} = B\Delta$ or $\dot{B} = \Delta B$ where $\Delta \in \mathfrak{g}$ is a constant matrix. In the next chapters we deal with gradient flows over matrix Lie groups.

4.5 Classic Matrix Lie groups

Here we briefly introduce the main matrix Lie groups encountered in this thesis:

4.5.1 The General Linear Group GL(n)

In fact GL(n) is the "biggest" matrix Lie group and all other matrix Lie groups are its subsets. It is a non-compact Lie group of dimension n^2 and its Lie algebra $\mathfrak{gl}(n)$ is $\mathbb{R}^{n\times n}$.

4.5.2 The Special Linear Group SL(n)

SL(n) is the group of all $n\times n$ matrices with unity determinant. It is a non-compact Lie group of dimension n^2-1 and its Lie algebra is $\mathfrak{sl}(n) = \{\Delta \in \mathbb{R}^{n\times n} | tr(\Delta) = 0\}$ where $tr(.)$ is the matrix trace operator. To see the relevance of this result we cite the identity $\det(e^{(t\Delta)}) = e^{tr(\Delta)t}$.

4.5.3 The Orthogonal Group O(n)

O(n) is the group of orthogonal $n \times n$ matrices. It is a compact Lie group of dimension $\frac{n(n-1)}{2}$ and its Lie algebra is $\mathfrak{o}(n) = \{\Delta \in \mathbb{R}^{n\times n} | \Delta = -\Delta^T\}$, i.e. the space of $n \times n$ skew-symmetric matrices. O(n) has two components: SO(n) which is the Lie group of orthogonal matrices with unity determinant and the other part is the set of orthogonal matrices with -1 determinant which is not a group.

There are other matrix Lie groups that we will consider in Chapter 6, groups such as non-singular diagonal or lower triangular matrices. All the above Lie groups can be equipped with suitable Riemannian metrics that match their group structure. As we will see in next chapters we can also derive gradient flow for minimization of a cost functions over these matrix groups.

Chapter 5

The Orthogonal Joint Diagonalization Gradient Flow

In this chapter we derive the gradient flow for the Orthogonal Joint Diagonalization problem, and consider its convergence properties. We also give another version of the JADE algorithm based on discretization of the gradient flow.

5.1 Introduction

Let us consider the set of $n \times n$ symmetric matrices $\{C_{i0}\}_{i=1}^{N}$. Define the cost function:

$$J_1(\Theta) = \sum_{i=1}^{n} \left\| \Theta C_{i0} \Theta^T - \mathrm{diag}(\Theta C_{i0} \Theta^T) \right\|_F^2 \tag{5.1}$$

where Θ belongs to the group of $n \times n$ orthogonal matrices, i.e. $\mathrm{O}(n)$. By minimizing this function over $\mathrm{O}(n)$ we will find the joint orthogonal diagonalizer of $\mathcal{C} = \{C_{i0}\}_{i=1}^{N}$. If $\{C_{i0}\}_{i=1}^{N}$ have an orthogonal joint diagonalizer this minimum will be zero, otherwise it is not zero. Yet, because of compactness of $\mathrm{O}(n)$ as it was

mentioned in Chapter 3 we are sure that a minimizer exists.

We follow and generalize the work of [Brockett] and [Helmeke] in developing gradient flows on $O(n)$ for this cost function.

5.2 The Gradient Flow for Minimization of $J_1(\Theta)$

Consider the Lie group $\mathrm{O}(n)$, at the point $\Theta \in \mathrm{O}(n)$ and for two tangent vectors ξ and η we define a Riemannian metric as:

$$\langle \xi, \eta \rangle_{\Theta} = \mathrm{tr}((\Theta^T \xi)^T \, \Theta^T \eta) = \mathrm{tr}(\xi^T \eta) \tag{5.2}$$

where tr(.) is the usual matrix trace operator. Based on this inner product we then find the gradient flow for the cost function J_1.

Theorem 5.1 *Consider the cost function $J_1(\Theta)$ then its gradient with respect to the Riemannian metric (5.2) is:*

$$\nabla J_1(\Theta) = -2\,\Theta \sum_{i=1}^{N} \left[\Theta^T \{\mathrm{diag}(\Theta C_{i0} \Theta^T)\Theta, C_{i0}\right] = -2 \sum_{i=1}^{N} \left[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\right] \Theta \tag{5.3}$$

where $[X, Y] = XY - YX$ is the Lie Bracket matrix. Moreover the gradient flow for minimization of this cost function is:

$$\dot{\Theta} = -\frac{1}{2}\nabla J_1(\Theta) = \sum_{i=1}^{N} \left[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\right] \Theta \tag{5.4}$$

Proof: In the Appendix.

A legitimate question is about the behavior of the solutions of this gradient system as $t \to +\infty$, that whether Θ converges at all or in the case that it converges to a single point, does it converge to a local or global minimum?. Using the La Salle's invariance principle (Section 4.3) observe that the equilibria of (5.4) $\Omega =$

$\{\Theta \in O(n) | \sum_{i=1}^{N} \left[\text{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\right] = 0\}$ is exactly the largest invariant set for which $\dot{J} = 0$. Hence any solution stays on O(n) and converges to Ω. Still this does not give any information about the nature of the convergence, i.e. whether it is single point-convergence or set-convergence. We state a theorem from [Mahony] that guarantees point-convergence for (5.4). This theorem considers gradient flows for analytic functions, i.e. functions that have Taylor expansion in a local coordinate around a point. In fact gradient flows of analytic functions have simple behavior, which is described in this theorem:

Theorem 5.2 *Let $J : M^n \to \mathbb{R}$ on the smooth Riemannian manifold M be an analytic function. Let the Riemannian metric at $x \in M$ in a local coordinate have a symmetric matrix representation Q_x. Assume that for any point $x \in M$ there exists an open neighborhood $U_x \subset M$ and $0 < \lambda_{1x} < \lambda_{2x}$ such that for all $z \in U_x$ we have $\lambda_{1x} I_{n\times n} \leq Q_z \leq \lambda_{2x} I_{n\times n}$, where $A_{n\times n} \geq B_{n\times n}$ means that $A - B$ is positive semi-definite. Then for $x(0) \in M$ in the gradient flow $\dot{x} = -\nabla J(x)$, either $x(t)$ leaves M or $\lim_{t\to+\infty} x(t) = x^*$ where x^* is a stationary(critical) point of J.*

Proof: See [Mahony].

Applying this theorem to the gradient flow (5.4), and noting the fact that J_1 is analytic on O(n), we conclude that for any initial condition there exists a $\Theta_\infty \in$ O(n) such that $\lim_{t\to+\infty} \Theta(t) = \Theta_\infty$ and Θ_∞ is a stationary point of J_1. Moreover, in general, we expect that the flow will converge to a local minimum, because in a generic case the Hessian matrix is invertible and hence all local minima are asymptotically stable and all other critical points are unstable.

With regard to global minimality of the answer we have no proof, but we have this conjecture:

Conjecture: In the generic case (for example when all the matrices are generated

randomly) all the local minimizers of $J_1(\Theta)$ where $\Theta \in \mathrm{O}(n)$are global and they are unique up to a row permutation.

Note that the above conjecture for $N = 1$ is equivalent to the fact for a symmetric matrix with distinct eigenvalues orthogonal diagonalizers are unique up to row permutation.

Assuming the above conjecture we can expect that if $\Theta(0)$ is not a stationary point then the solution will converge to a global minimum. Extensive simulations support this.

5.3 The Double Bracket Equation

We can also find differential equations that govern the evolution of the matrices under diagonalization $\{\Theta C_{i0}\Theta^T\}_{i=1}^N$. We consider the original matrices as initial conditions and define $C_i(t) = \Theta(t)C_{i0}\Theta^T(t)$ as the matrices under diagonalization in time. Note that we require $\Theta(0) = I_{n\times n}$ so that $C_i(0) = C_{i0}$. The next theorem gives a set of differential equations known as Double Bracket equations [Helmke] that describe the evolution of $C_i(t)$ in time.

Theorem 5.3 *Let $\Theta(t) \in O(n)$ with $\Theta(0) = I_{n\times n}$ satisfy the gradient flow for minimizing J_1 as in the previous theorem, then each of the matrices under diagonalization, i.e. $C_j(t) = \Theta(t)C_{j0}\Theta^T(t)$ satisfies:*

$$\dot{C}_j = \left[\sum_{i=1}^{N}[\mathrm{diag}(C_i), C_i], C_j\right] \qquad 1 \le j \le N \tag{5.5}$$

Proof: In the Appendix.

The Double Bracket equation can be shown to be a gradient flow of the cost function $f(\{C_i\}_{i=1}^N) = \sum_{i=1}^n \left\|C_i - \mathrm{diag}(C_i)\right\|_F^2$ with respect to a Riemannian metric known as Normal Riemannian metric over the manifold $M(\mathcal{C}) = \{(C_1, ..., C_N)|C_i =$

$\Theta C_{i0}\Theta^T,\ \Theta \in \mathrm{O}(n)\}$ [Helmke]. We follow [Helmke] to show this fact. Let $C \in M(\mathcal{C})$, then we can show that the tangent space at $C = (C_1, ..., C_N)$ to $M(\mathcal{C})$ is $T_C M(\mathcal{C}) = \{([\Omega, C_1], ..., [\Omega, C_N]) | \Omega \in \mathfrak{o}(n)\}$, where $\mathfrak{o}(n)$ is the set of $n \times n$ skew-symmetric matrices, i.e the Lie algebra of $\mathrm{O}(n)$. Obviously the mapping $\Omega \in \mathfrak{o}(n) \mapsto ([\Omega, C_1], ..., [\Omega, C_N])$ is a linear map that is not 1-1 over its image $T_C M(\mathcal{C})$. The kernel of this map is $K = \{\Omega \in \mathfrak{o}(n) | ([\Omega, C_1], ..., [\Omega, C_N]) = 0\}$. The orthogonal complement of K, defined as $K^\perp = \{\Omega \in \mathfrak{o}(n) | \mathrm{tr}(\Omega^T \Delta) = 0,\ \forall \Delta \in K\}$ is in 1-1 correspondence with $T_C M(\mathcal{C})$ and any $\Omega \in \mathfrak{o}(n)$ can be written as $\Omega = \Omega^{\|} + \Omega^\perp$ where $\Omega^{\|} \in K$ and $\Omega^\perp \in K^\perp$. A useful observation is that for any symmetric matrix $H_{n\times n}$, the matrix $H_C = ([H, C_1], ..., [H, C_N])$ belongs to $K^\perp$, because $\mathrm{tr}(H_C^T \Omega) = \sum_i \mathrm{tr}(H[\Omega, C_i]) = 0$ for any $\Omega \in K$. In the next step we define an inner product between $\xi_1, \xi_2 \in T_C M(\mathcal{C})$ in terms of pre-images of ξ_1, ξ_2 in $K^\perp$, indeed we can define the Normal Riemannian metric as:

$$\langle \xi_1, \xi_2 \rangle_C = \mathrm{tr}(\Omega_1^{\perp T} \Omega_2^\perp)$$

where $\Omega_1^\perp, \Omega_2^\perp \in K^\perp$ are such that $\xi_i = ([\Omega_i, C_1], ..., [\Omega_i, C_N])$ for $i = 1, 2$. Note that for any $\Omega^\perp \in K^\perp$ the directional derivative of f along $\xi = ([\Omega^\perp, C_1], ..., [\Omega^\perp, C_N])$ is given by $Df|_\xi = -\mathrm{tr}(\Omega^{\perp T} 2 \sum_{i=1}^N [\mathrm{diag}(C_i), C_i])$. By the properties of the gradient vector with respect to the Normal Riemannian metric we can see that $\nabla f = ([X, C_1], ..., [X, C_N])$ for some $X \in K^\perp$ such that $Df_\xi = \mathrm{tr}(\Omega^\perp X)$, therefore $X = -2 \sum_{i=1}^N [\mathrm{diag}(C_i), C_i]$. So we state:

Theorem 5.4 *The flow:*

$$(\dot{C}_1, ..., \dot{C}_N) = -\frac{1}{2} \nabla f = \left([\sum_{i=1}^N [\mathrm{diag}(C_i), C_i], C_1], ..., [\sum_{i=1}^N [\mathrm{diag}(C_i), C_i], C_N] \right)$$

which is exactly the flow in (5.5) is a gradient flow for minimization of f on $M(\mathcal{C})$ with respect to the Normal Riemannian metric defined above.

5.4 Discretization of the Gradient Flow

The subject of discretization of differential equations on manifolds and groups is a new and challenging field, it is also known as *structure preserving integration* or *geometrical integration* [Devore]. The difficulty lies where we should keep the updated answer always on the manifold. For example, for (5.4) we should have updates that give a $\Theta_k \in \mathrm{O}(n)$ for any integer k. It is shown in [Calvo] that only implicit Runge-Kutta methods can give orthogonal updates, and in general an implicit Runge-Kutta is costly to solve and it is difficult to find exact solutions for. Thus we should expect to have methods that can retain orthogonality only approximately, this is in contrast to the JADE algorithm or Jacobi method that retains orthogonality by construction.

In the sequel we will use the Euler and fourth order (Explicit) Runge-Kutta discretization methods to discretize (5.4). The Euler method is very simple and because of its immediate relation to the gradient descent method is considered. The Euler and Runge-Kutta methods have no measures to keep the updates on $\mathrm{O}(n)$, although by small enough step-size we can achieve almost orthogonal answers. We will use fixed-step size schemes mainly because they are simple to implement and that we require small step-size more than anything for having almost orthogonal answer. We also briefly introduce two other methods, *the adjoint equation method* [Calvo] and *Cayley-transform methods*[Iserles] that try to retain orthogonality in updates, but are more complicated and computation demanding.

5.4.1 The Euler Discretization Method

The Euler method or the first order Runge-Kutta method to discretize (5.4) results in:

$$\Theta_{k+1} = \big(I - \mu_k \sum_{i=1}^{N} \big[\mathrm{diag}(\Theta_k C_{i0} \Theta_k^T), \Theta_k C_{i0} \Theta_k^T\big]\big)\Theta_k = \big(I - \mu_k \Delta_k\big)\Theta_k \quad k \geq 0 \quad (5.6)$$

where μ_k is the step size for the k^{th} update and Θ_0 is an orthogonal initial condition. In (5.6), Δ_k due to the definition of the Lie bracket is a skew-symmetric matrix, so if we assume Θ_{k-1} is orthogonal we have:

$$\mathcal{D}_k = \Theta_k \Theta_k^T - I = I - \mu_k(\Delta_k + \Delta_k^T) + \mu_k^2 \Delta_k \Delta_k^T - I = \mu_k^2 \Delta_k \Delta_k^T$$

where $\mathcal{D}_k$ is called *local orthogonality error*, given the assumption that Θ_{k-1} is orthogonal. We can see that the local orthogonality error is of the second order in μ_k. It should be noted that the Euler method is exactly the same as gradient descent method, so at each step for small enough step size the cost function can be reduced by this method.

One important issue is how to choose the step size μ_k at each step so that we have stable (bounded) and fast converging answer which has small orthogonality error. In numerical optimization it is usually desired to find the largest step size that gives the largest reduction in the function value, on the other hand here we also have the orthogonality constraint which requires small step sizes. Therefore we encounter two opposite requirements in choosing the step size. It seems that algorithms to find the best step size (fulfilling both mentioned opposite goals) will be very costly. Hence in practice we try to find a small and constant step size that achieves both goals. Evidently this step size depends on many factors, such as the dimension of the matrices, the number of matrices and the relative scaling of the matrices. In fact almost all numerical optimization and integration methods

Algorithm 5.1

1. **Set** μ **and** ϵ.
2. **Set** $\Theta_0 = I_{n\times n}$ or to a "good" initial guess.
3. **While** $\|\Delta_k\|_F > \epsilon$ **do**

$\Theta_{k+1} = \big(I - \mu\Delta_k\big)\Theta_k$

if $\|\Theta_{k+1}\|_F$ **or** $\|\Theta_{k+1}\Theta_{k+1}^T - I_{n\times n}\|_F$ are *"big"* **then** *reduce* μ **and goto** 2.

4. **End**

Table 5.1: A fixed-step size implementation of the Euler discretization of the gradient flow (5.4), which is re-written in the form $\dot{\Theta} = \Delta_{\Theta(t)}\Theta(t)$. The unspecified parameters and qualities are to be decided in practice.

require some step size tuning that depend on the nature of the data and should be found by trial and error. On the other hand it is rather easy to find out if the chosen step size is bad, i.e. if $\|\Theta_k\|_F$ is becoming "big" or far from orthogonal then this means that the guessed μ is not appropriate. Therefore we ought to reduce μ and re-start the algorithm. Of course, it is possible to devise more elaborate adaptive methods.

Table (5.1) depicts this algorithm in pseudo code (see equation (5.6)).

Remark: In the context of BSS a "good" initial guess for Θ_0 is in fact the transpose of an eigen matrix corresponding to any single cumulant slice.

In the above code there are still some un-specified things such as how to choose the initial μ and ϵ and what do we mean by "big" and how to reduce μ in case of observing instability or large orthogonality error. These are issues that depend on the data at hand and the desired accuracy in the solution. In Section 5.6 we consider the effect of these parameters in practice.

5.4.2 Runge-Kutta (RK) Methods

Runge-Kutta methods are well known numerical integration methods [Stuart]. Here we first introduce the general form of Runge-Kutta methods for discretization of $\dot{X} = f(X)$, where $X(t) \in \mathbb{R}^{n\times n}$ and $f : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}^{n\times n}$ is a smooth enough vector field. Then the classical s stage (RK) method is written as:

$$Y_i = X_k + \mu \sum_{j=1}^{s} a_{ij} f(Y_j), \quad i = 1, ..., s \tag{5.7}$$

$$X_{k+1} = X_k + \mu \sum_{i=1}^{s} b_i f(Y_i), \quad X_0 = X(0) \tag{5.8}$$

where μ is called step-size, Y_i's are called the stage-values. Numbers a_{ij} form a matrix A and b_i's form a vector $\vec{b}$. b is such that $\sum_{i=1}^{s} b_i = 1$. If $a_{ij} = 0$ for $i < j$ then the RK scheme is called *explicit* otherwise it is called *implicit*. The order of a RK method is related to the accuracy of the first step error $E_1 = X(\mu) - X_1$, that is the method is of r^{th} order if $E_1 = F(X_0)\mu^{r+1} + O(\mu^{r+2})$, for some matrix function F where $\|O(\mu^{r+2})\|_F \leq K\mu^{r+2}$ for small h and a constant matrix K. Hence an s stage method is not necessarily of order s. Note that the Euler method described above is a one stage and first order method.

It should be noted that the above formulation is a vector space based formulation, so obviously if it is applied to a differential equation on a manifold again, the issue of staying on the manifold arises. In [Calvo] it is shown that no explicit RK method can retain orthogonality. Moreover in [Calvo] it is shown that if an r^{th} order method is applied for a differential equation of the form $\dot{\Theta} = F(\Theta)\Theta \quad (*)$ where $\Theta(0) \in \mathrm{O}(n)$, then the local orthogonality error defined as $\mathcal{D}_k = \Theta_k \Theta_k^T - I$ is such that $\|\mathcal{D}_k\|_F \leq L_k \mu^{r+1}$ for small enough μ and a constant L_k. Hence we should expect that a higher order method will result in a better orthogonality error for the same value of step-size. In [Calvo] a method of *adjoint equations* is introduced

that achieves the above bound for an explicit RK method of order $r-1$.

Another possible idea is to use the *Cayley transform*. The Cayley transform of $\Theta \in \mathrm{O}(n)$ is defined as: $\Omega = \mathrm{Cay}(\Theta) = \frac{1}{2}(I-\Theta)(I+\Theta)^{-1}$, it is a skew-symmetric matrix and belongs to $\mathfrak{o}(n)$ and the inverse Cayley transform given by $\Theta = (I-\frac{1}{2}\Omega)^{-1}(I-\frac{1}{2}\Omega)$ belongs to $\mathrm{O}(n)$. The Cayley-transform methods [Iserles] for discretization of (*) are based on the idea of solving (advancing) the counterpart of (*) on the Lie algebra at any point and again transforming the answer back to the group $\mathrm{O}(n)$. Note that because the Lie algebra is a linear space we have no difficulty in keeping the answer in it. For more detail the reader is referred to [Calvo].

One point to mention is that the underlying problem in our case is an optimization problem for which we derived a gradient descent ODE. In this context one may want to have the property that at each step of discretization the cost function is reduced, the same as standard gradient descent methods. As we mentioned the Euler method obviously has this property, that is for some μ_0, any $\mu \leq \mu_0$ results in an update that reduces the cost function and moreover the stationary points of the discretized equation and the discrete algorithm are the same. On the other hand whether the RK methods posses this property is not clear in general. In [Stuart p.519] it is shown that under the additional assumptions that the gradient of the cost function is globally Lipschitz and radially unbounded this is true for RK methods in $\mathbb{R}^n$. Hence we may expect this to hold to a good extent on other manifolds locally. Nevertheless, this form of discretization is not popular in the optimization community.

Algorithm 5.2

1. **Set** μ **and** ϵ.

2. **Set** $\Theta_0 = I_{n\times n}$ or to a good initial guess.

3. **While** $\|\Delta_k\|_F > \epsilon$ **do**

$Y_1 = \Theta_k$

$Y_2 = \Theta_k + \frac{1}{2}\mu f(Y_1)$

$Y_3 = \Theta_k + \frac{1}{2}\mu f(Y_2)$

$Y_4 = \Theta_k + \mu f(Y_3)$

$\Theta_{k+1} = \Theta_k + \mu\big(\frac{1}{6}f(Y_1) + \frac{1}{3}f(Y_2) + \frac{1}{3}f(Y_3) + \frac{1}{6}f(y_4)\big)$

if $\|\Theta_{k+1}\|_F$ **or** $\|\Theta_{k+1}\Theta_{k+1}^T - I_{n\times n}\|_F$ are "*big*" **then** *reduce* μ **and goto** 2.

4. **End**

Table 5.2: A fixed-step size implementation of the fourth order RK discretization of the gradient flow (5.4), which is re-written in the form $\dot{\Theta} = \Delta_\Theta\Theta = f(\Theta)$. The unspecified parameters and qualities are to be decided in practice.

Fourth Order RK Method Discretization for Θ

One of the most popular RK methods is the classical four-stage fourth order explicit RK method for which A is such that $a_{21} = \frac{1}{2}$, $a_{32} = \frac{1}{2}$, $a_{43} = 1$ and all other elements of A are zero and $\vec{b}$ is $[\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}]^T$. We use this scheme to discretize (5.4). We will use a fixed small step-size to implement this method. Again we re-write (5.4) in the form $\dot{\Theta} = \Delta_\Theta\Theta = f(\Theta)$. The derived algorithm is shown in Table (2).

5.5 Applications in the BSS/ICA Problem(EG-JADE and RKG-JADE Algorithms)

We can use the methods shown in Tables (5.1) and (5.2) to construct gradient based versions of the JADE algorithm, simply by replacing the Jacobi based joint diagonalization by the algorithms developed in previous section. We call the method using the Euler method Euler-Gradient based JADE or the EG-JADE algorithm and the other one Runge-Kutta Gradient based JADE or RKG-JADE. These two methods retain orthogonality in Θ only approximately. However, in the case of noisy ICA or when using estimated cumulants in the JADE algorithm, we know that after whitening the data the separating matrix is not an orthogonal one necessarily or exactly, therefore the constraint of keeping the update orthogonal can be compromised (see Section 7.1 for more detail). In other words for the ICA/BSS problem we can tolerate slightly off-O(n) answers of (5.4). Of course we will lose the compactness property of O(n) which is a theoretical guarantee for the convergence of flow (5.4).

5.6 Numerical Examples

In this section we examine the performance of Algorithms 5.1 and 5.2 in the context of joint diagonalization and ICA/BSS problems.

Example 1: In this example we generate $N = 100$ symmetric 20×20 ($n = 20$) matrices of the form $C_i = U\Lambda_i U^T, 1 \leq i \leq N$ where Λ_i are diagonal matrices with elements that have i.i.d standard Gaussian distribution. We find $\{C_i\}$'s joint diagonalizer using Algorithm 5.1. In the first set of experiments we set $\mu = .0001, .0002, .0006, .0012$ and run the algorithm to observe the behavior of

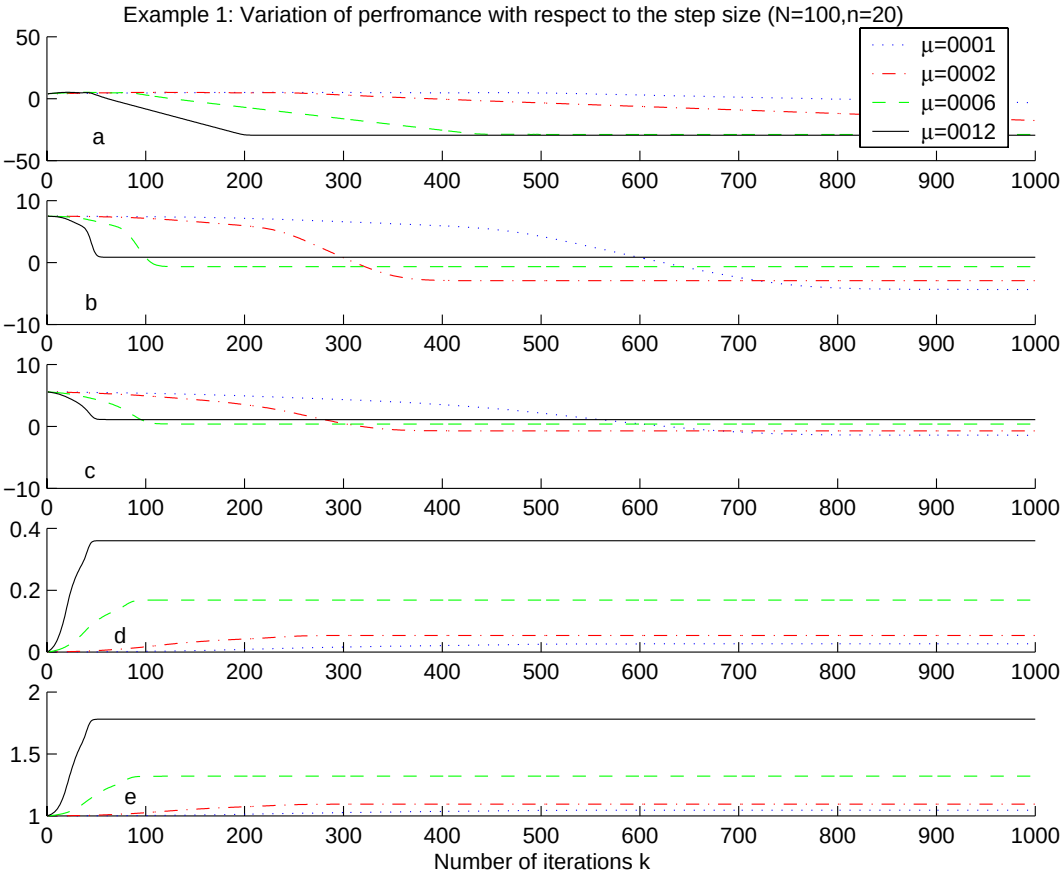

Figure 5.1: This figure shows variation of some performance measures computed in Example (1) for different values of step size μ: a. $\log(\|\nabla J_1(\Theta_k)\|)$ b. $\log(J_1(\Theta_k))$ c. $\log(Index(P_k))$ d.$\|\mathcal{D}_k\|$ e. $\det(\Theta_k)$

different quantities (we do not use any stop criteria, but run the iteration for k=1000 times). Figure (5.1) shows the variation of some quantities in terms of the number of iterations k. The first graph depicts $\log(\nabla J(\Theta_k))$. Because of the large dynamic range of the graph we used logarithm. Note that the gradient decreases much faster and deeply for large μ's. The second graph shows $\log(J_1(\Theta_k))$. The third graph shows how far $P_k = \Theta_k U$ is from being essentially diagonal, using the $Index$ criterion defined by equation (2.6). In fact it shows $\log(Index(P_k))$. The

ideal value for $J_1(\Theta_\infty)$ and $Index(P_\infty)$ is zero. So obviously the smaller values of μ result in better performance although the corresponding gradients are not smaller in norm. The reason will become clear by looking at the next two graphs where they show measures of orthogonality. The fourth graph shows $\|\mathcal{D}_k\|$ and the fifth one shows $\det(\Theta_k)$. These two graphs show that the smaller values of μ result in better orthogonality error and so that is why they give better performance. Among these values for μ it seems that $\mu = .0006$ is a good one that gives a fair balance between computational complexity and performance.

In the next experiment we increase the dimension and number of matrices. Figure (5.2) shows the results. In this figure we have four set of curves. We consider $N = 100, n = 40$ with $\mu = .0003, .0001$ and $N = 200, n = 40$ with $\mu = .0003, .0001$. It is interesting to note that the convergence by increasing n, N or μ will be faster however the performance will deteriorate, because the orthogonality error increases.

We should mention that the matrices generated in the above experiments were somehow "well" scaled, in the sense that Λ_i's generated have almost the same dynamic range of values and this made the convergence of the algorithm much better. Although this may not be the case in a general JD problem, we can argue that in the BSS/ICA problem after whitening the data because the correlation matrix is identity, the dynamic range of the data is restricted so the cumulant slices will have almost close norms. As an example for the case that bad scaling makes things difficult we consider the first experiment $N = 100, n = 20$ but we consider $C_i = \sqrt{i}\ U\Lambda_i U$ then algorithm (5.1) does not converge for $\mu = .0001$ at all. Even by decreasing μ to .000001 still the algorithm does not converge. Hence in the case that the dynamic range of the matrices are different this algorithm will not work

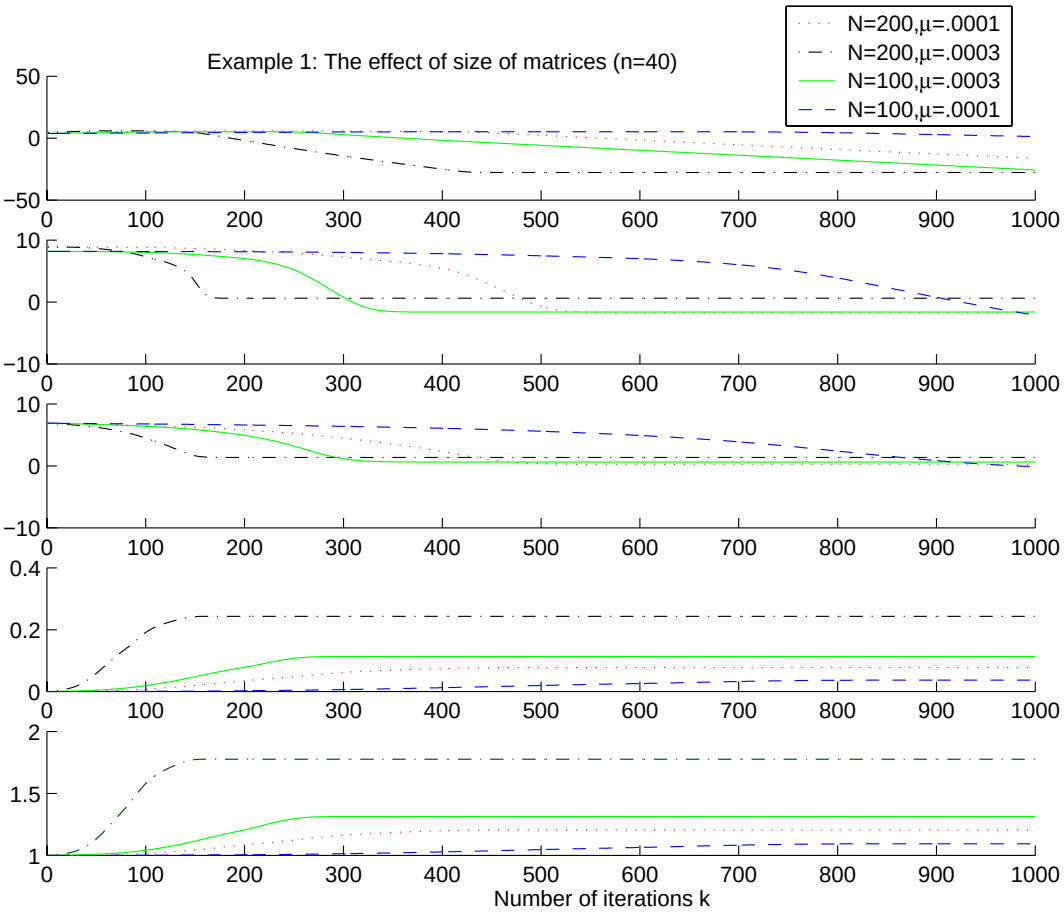

Figure 5.2: This figure shows how the performance changes when the dimension and number of matrices is increased in Example (1). $N = 100, n = 40$ and $N = 200, n = 40$ are used with $\mu = .0003, .0001$. a. $\log(\|\nabla J_1(\Theta_k)\|)$ b. $\log(J_1(\Theta_k))$ c. $\log(Index(P_k))$ d.$\|\mathcal{D}_k\|$ e. $\det(\Theta_k)$

properly, although in the case of BSS problem after whitening this is unlikely to be the case.

Example 2: In this example we apply the EG-JADE algorithm to a BSS problem. In the first experiment we consider $n = 4$ independent sources $\vec{\mathbf{s}}$ mixed through a randomly generated mixing matrix:

$$A = \begin{bmatrix} -1.72577265692312 & 0.13866495543565 & 1.15075435309812 & -0.37346107032557 \\ 0.81319959967304 & -0.85953392675766 & -0.60802501127072 & -0.83203043468849 \\ 1.44186661829232 & -0.75225055816553 & 0.80615791615996 & 0.28686630098359 \\ 0.67227220216042 & 1.22961508382908 & 0.21713285248002 & -1.81889162356406 \end{bmatrix}$$

One of the sources has exponential distribution with zero mean and parameter $\lambda = 1$, the next one has two-sided exponential distribution (also known as Laplace distribution) with parameter $\lambda = 1$, the other two sources have uniform distribution on $[-\frac{1}{2}, \frac{1}{2}]$. We generate $T = 2000$ sample of the source data denoted by the matrix S_T (it is a $4 \times T$ matrix), then mix the data to get the mixed signal samples $X_T = AS_T$. We then zero the mean of each row of X_T and then compute $R_{\mathbf{xx}} = \frac{1}{T} X_T X_T^T$. In the next step we find $R_{\mathbf{xx}}^{-\frac{1}{2}}$, using the eigen decomposition of $R_{\mathbf{xx}}$ and whiten the data to get $Y_T = R_{\mathbf{xx}}^{-\frac{1}{2}} X_T$[1]. Then using the sample estimates of cumulants we compute the cumulant tensor, and thereafter $\mathcal{C}_{\mathbf{y}} = \{Cum_{\mathbf{y}}(:,:,i,j) | 1 \leq i,j \leq n = 4\}$. Note that due to the symmetry of cumulant tensor we do not need to compute all the cumulants, and in fact $Cum(:,:,i,j) = Cum(:,:,j,i)$. Next we pass the set $\mathcal{C}_{\mathbf{y}}$ through the EG-JADE algorithm (Algorithm 5.1) with parameters $\mu = .01$ and $\epsilon = .1$. We do not use any initial guess such as an eigen matrix of one of cumulant slices. This could enable

[1]Direct computation of $R_{\mathbf{xx}}$ and afterwards $R_{\mathbf{xx}}^{-\frac{1}{2}}$ is not the best practical way to whiten the data both from computational load and accuracy points of view, however, for our purposes and due to the good accuracy of computations in $MATLAB^{®}$ this method is satisfactory. For more details the reader can consult [Comon 1] and the reference therein.

faster convergence. The algorithm stops after $k = 95$ iterations yielding:

$$\Theta_{95} = \begin{bmatrix} 0.95985004288776 & -0.28044871233207 & 0.00550000868509 & 0.03592182236959 \\ 0.27822003344298 & 0.94999801900180 & 0.14445051912210 & -0.03875257025448 \\ -0.02276020813643 & -0.13853724807153 & 0.78968278035147 & -0.63503749191227 \\ -0.04603839531831 & -0.04412894967835 & 0.63399928692651 & 0.79979176271809 \end{bmatrix}$$

with $\|\Theta_{95}\Theta_{95}^T - I_{4\times4}\|_F = 0.06586700007617$. The estimated un-mixing matrix $B = \Theta_{95}R_{\mathbf{xx}}^{\frac{-1}{2}}$ is such that the the total mixing matrix is :

$$P = BA = \begin{bmatrix} 0.03884229547725 & 0.01947545028173 & \mathbf{3.42035143547487} & 0.09397388342959 \\ 0.02977653938651 & -0.01591949619921 & -0.16913582939640 & \mathbf{3.50898334470220} \\ 0.05539773531559 & \mathbf{-0.72804272424862} & 0.04057529899273 & 0.05829470954786 \\ \mathbf{-0.99678892890767} & -0.00922039815672 & 0.08732420762586 & 0.10336859023621 \end{bmatrix}$$

Note that it is close to a permuted diagonal matrix and in fact its distance from being essentially diagonal is $Index(P) = 0.86359255677199$. We can give two reasons that $Index(P)$ is not exactly zero: first, we are using estimates of cumulants rather than exact values; second, we have not proceeded further to minimize the cost function (the effect of $\nabla J_1(\Theta_{95})$ not being exactly zero) and that we have orthogonality error. However, the first cause is more harmful. In fact with $\epsilon = .0001$ and $\mu = .0001$ we can have $Index(P) = 0.74485563657809$ after $k = 35056$ iterations!. Here we can notice a drawback rampant among gradient based optimization methods (as opposed to Newton based methods), which is their slow convergence when they are close to the answer.

To better understand the effect of mixing and un-mixing on the data we plot components of the source, sensed, whitened and un-mixed data with respect to each other in order to visualize their dependency. Figure (5.3) shows these graphs. From the boundaries, especially in the case of uniform variables, we may understand the dependency. The first row shows the independent source data. The second row shows the mixed data. The third row shows the whitened data, we can see that still some dependency has remained. The last row shows almost independent data.

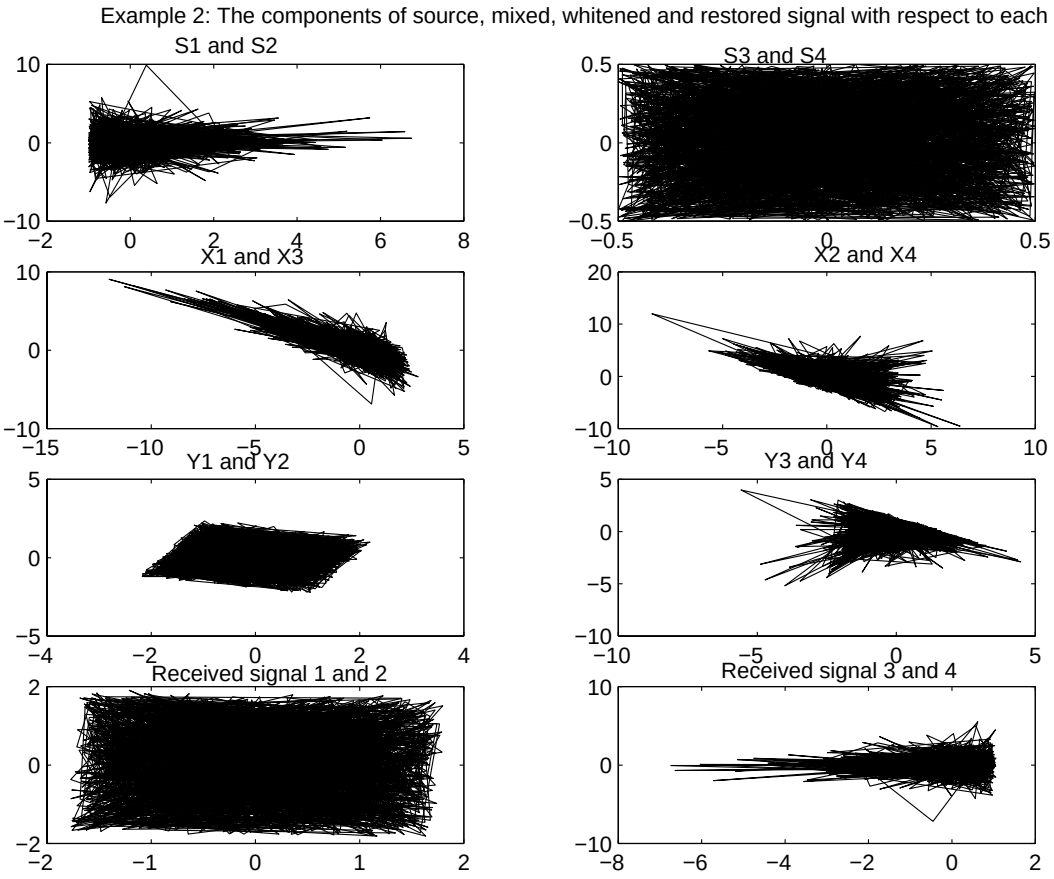

Figure 5.3: This figure tries to help to visualize the dependence between the components of data during each step of the mixing-unmixing process in the first experiment Example (2). Each graph is one component of data vector with respect to another component. a) $\mathbf{s}_1$ and $\mathbf{s}_2$, they are one-sided and two-sided exponentials, respectively. They are independent. b) $\mathbf{s}_3$ and $\mathbf{s}_4$ and both are uniform. They are independent. c) $\mathbf{x}_1$ and $\mathbf{x}_3$. They are dependent. d) $\mathbf{x}_2$ and $\mathbf{x}_4$. They are dependent. e) $\mathbf{y}_1$ and $\mathbf{y}_2$. They are uncorrelated but still dependent. f) $\mathbf{y}_3$ and $\mathbf{y}_4$. They are uncorrelated but still dependent. g) $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$. They are almost independent($\vec{\hat{\mathbf{s}}} = BA\vec{\mathbf{s}} = P\vec{\mathbf{s}}$). h) $\hat{\mathbf{s}}_3$ and $\hat{\mathbf{s}}_4$. They are almost independent.

Note that the element with maximum absolute value in the last row of P (this corresponds to the recovered one-side exponential source) has negative sign, that is why the graphs (a) and (h) are reverse of each other.

In the next step we consider the effect of changing μ on the convergence rate and separation performance of the algorithm. The table below shows the results.

We can see that there is not much improvement in the separation performance by

	k	$Index(P)$	$\|D_k\|_F$
$\mu = .04$	DNC	-	-
$\mu = .03$	31	0.90465915694489	0.02068842693575
$\mu = .01$	95	0.86359255677199	0.00229871410397
$\mu = .005$	191	0.85442048722669	5.746785259929237e-004
$\mu = .001$	959	0.84746473734464	2.298714103972716e-005

Table 5.3: This table gives the performance measures from applying the source separation algorithm developed in this chapter for the sources and mixing matrix A in Example (2) with respect to the step size μ. Note that for $\mu = .04$ the algorithm does not converge.

decreasing μ, however orthogonality error is reduced. As we can see improvement in orthogonality error does not improve the separation performance considerably. We can conclude that the Euler discrete JD scheme with moderately small step-size results in acceptable separation performance with low computational cost (compared to smaller step-sizes). Note that the algorithm does not converge for $\mu = .04$ and in fact Θ_k blows up very fast. This can be related to the fact that μ is not small enough to guarantee reduction in the cost function at each step.

Example 3: In this example we will compare the performance of the EG-JADE

and RKG-JADE algorithms with the JADE algorithm[2]. We have no indication that these methods can perform better than JADE, because they are different versions of each other. In fact because it retains the orthogonality by construction and because of its almost quadratic convergence rate [Bunse-Gerstner], the JADE algorithm outperforms this gradient based method slightly in separation performance and largely in computational cost.

Consider the data model:

$$\vec{\mathbf{x}} = A\vec{\mathbf{s}} + \sigma\vec{\mathbf{n}}$$

where $\vec{\mathbf{n}}$ is the standard Gaussian noise and σ^2 indicates the noise power. We compare the performance of EG-JADE and RK-JADE with the standard JADE's performance for the same mixing matrix A and source signals as in Example 1. We compute the averaged performance measure $\overline{Index(P)}$ (see (2.6)) versus σ in the model for different values of $T = 2000, 5000$, $\epsilon = .01$ and $\mu = .01$ for these algorithms. To calculate the average of $Index(P)$ we repeat each experiment 100 times for any set of the above parameters.Figure (5.4) shows the results. From Figure (5.4) we can see that the three algorithms perform almost equally up to moderate values of σ. For larger noise power the JADE algorithm slightly outperforms the other two. This result is somehow un-expected. This can mean that the cost function $J_1(\Theta)$ is difficult to be minimized by gradient descent when the matrices are not jointly diagonalizable. It is interesting to note that the Euler and RK methods also have the same performance in this problem.

[2]A $MATLAB^{®}$ code for JADE was downloaded from: http://tsi.enst.fr/čardoso/icacentral/Algos

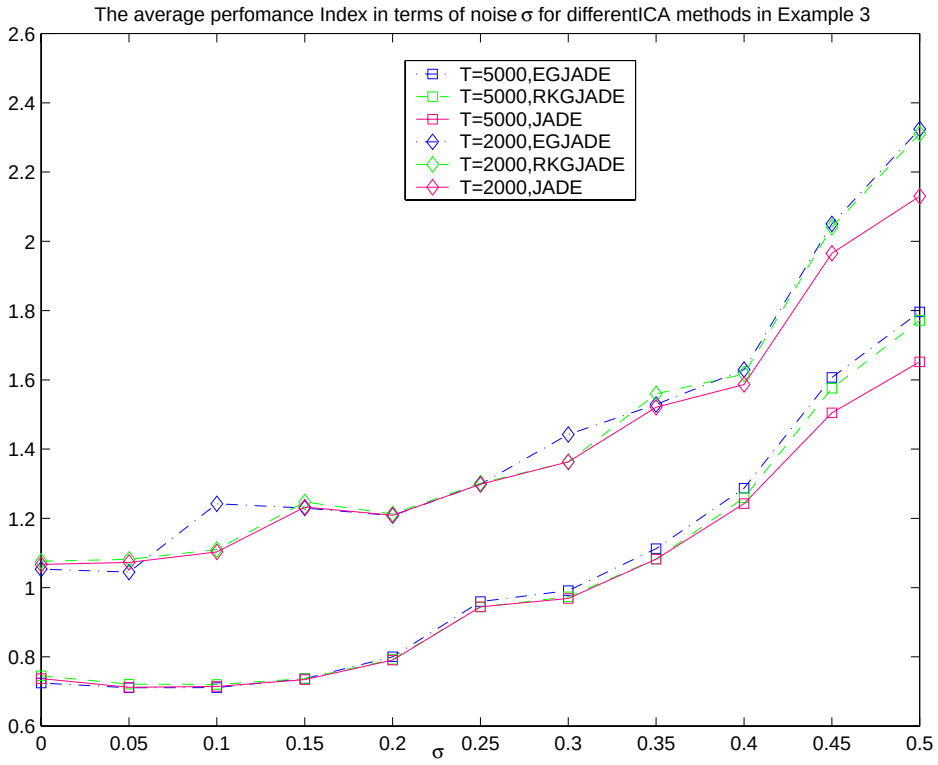

Figure 5.4: This figure shows the average performance index of EG-JADE and RKG-JADE with $\epsilon = \mu = .01$ and also average performance index of JADE for $T = 2000, 5000$. The mixing matrix and sources are the same as in the first part of Example (2). Note that the graphs are bundled in two groups corresponding to three different values of T.

Chapter 6

Gradient Based Non-Orthogonal Joint Diagonalization

In this chapter we derive gradient based algorithms for Joint Diagonalization by non-orthogonal matrices using the cost function J_1 introduced in Chapter 3. We develop methods to discretize them, such that the answer has some desired properties that resemble the continuous answer. We also give a method based on the Armijo Step Size selection to implement the gradient flow.

6.1 Gradient Flow for Joint Diagonalization on $\mathbf{GL}(n)$

Let $\{C_i\}_{i=1}^N$ be a set of $n \times n$ symmetric matrices and consider the cost function introduced in (3.5) as:

$$J_1(B) = \sum_{i=1}^{N} \left\| BC_iB^T - \operatorname{diag}(BC_iB^T) \right\|_F^2 \tag{6.1}$$

where the un-mixing matrix B belongs to $\mathrm{GL}(n)$, the group of non-singular $n \times n$ matrices. As it was mentioned in Section 3.3, $J_1(B)$ is not scale-invariant, i.e. $J_1(\Lambda B) \neq J_1(B)$ for non-singular diagonal Λ and hence it is not suitable for JD. We remind that scale-invariance for a cost function in terms of un-mixing matrix refers to left multiplication of the argument by diagonal matrices. Note that as $\|\Lambda\|_F \to 0$, $J_1(\Lambda B)$ decreases to zero. In the next two sections we will develop methods to avoid such reductions.

Consider the manifold $\mathrm{GL}(n)$, at any point $B \in \mathrm{GL}(n)$ for tangent vectors $\xi, \eta \in T_B^{GL(n)}$ we define a Riemannian inner product as:

$$\langle \xi, \eta \rangle_B = \mathrm{tr}((\xi B^{-1})^T \eta B^{-1}) = \mathrm{tr}(B^{-T} \xi^T \eta B^{-1}) = \mathrm{tr}(\eta (B^T B)^{-1} \xi^T) \tag{6.2}$$

In fact $Q_B = (B^T B)^{-1}$ is the positive definite matrix representing the Riemannian inner product at the point B. Note that ξB^{-1} belongs to the Lie Algebra of $\mathrm{GL}(n)$, i.e $\mathfrak{gl}(n)$. We call this Riemannian metric the *Natural Riemannian metric*[1], because it matches the group structure of $\mathrm{GL}(n)$. Note that we could take $\langle \xi, \eta \rangle_B = \mathrm{tr}((B^{-1}\xi)^T B^{-1}\eta)$ as the Riemannian metric, however, because we are concerned about left multiplication of B by diagonal matrices we are interested in tangent vectors of the form ΔB where $\Delta \in \mathfrak{gl}(n)$.

The gradient flow for minimization of (6.1) on $\mathrm{GL}(n)$ corresponding to the Natural Riemannian metric is given in the following theorem:

Theorem 6.1 *:The Natural Gradient of (6.1) on GL(n) is*

$$\nabla J_1(B) = 4\Big(\sum_{i=1}^{N} \big(BC_iB^T - \mathrm{diag}\big(BC_iB^T\big)\big) BC_iB^T \Big) B \tag{6.3}$$

[1]We use this name after Amari's Natural Gradient method which uses the same metric (see Chapter 2 in [Haykin]).

and the Natural gradient flow for minimizing (6.1) on GL(n) is:

$$\dot{B} = -\Big(\sum_{i=1}^{N} \big(BC_iB^T - \mathrm{diag}\big(BC_iB^T\big)\big)BC_iB^T \Big)B \tag{6.4}$$

with $B(0) \in$ GL(n).

Proof: See the Appendix.

The equilibria of the above flow is found by letting $\nabla J_1(B) = 0$ or $H = \sum_{i=1}^{N} \big(BC_iB^T - \mathrm{diag}\big(BC_iB^T\big)\big)BC_iB^T = 0$. Therefore:

$$\mathrm{tr}(H) = \mathrm{tr}\Big(\sum_{i=1}^{N} \big(BC_iB^T - \mathrm{diag}(BC_iB^T)\big)\big(BC_iB^T - \mathrm{diag}(BC_iB^T) + \mathrm{diag}(BC_iB^T)\big) \Big) =$$

$$\sum_{i=1}^{N} \mathrm{tr}\big(\big(BC_iB^T - \mathrm{diag}(BC_iB^T)\big)^2\big) = 0$$

where we used the fact that $\mathrm{tr}((A - \mathrm{diag}(A))\mathrm{diag}(A)) = 0$ for any A. The above result means that $BC_iB^T = \mathrm{diag}(BC_iB^T)$ for all $1 \leq i \leq N$. Hence if the set $\{C_i\}_{i=1}^N$ are not diagonalizable, i.e. there is no non-singular B such that all $\{BC_iB^T\}_{i=1}^N$ are diagonal then, this flow does not have any equilibria or $J_1(B)$ does not have any minima on GL(n)!. This can be related to the fact that $J_1(B)$ can be reduced to zero by a diagonal matrix whose norm is approaching zero and again shows that $J_1(B)$ by its own is not a good cost function for joint diagonalization, as mentioned before.

6.2 Gradient Flow for Joint Diagonalization over SL(n)

One way to improve the problem with non-compactness of GL(n) and the scale variability of $J_1(B)$ is to consider minimization of $J_1(B)$ over the set SL(n), i.e.

the group of $n \times n$ matrices with unit determinant. Obviously SL(n) is not a compact group and $\det(B) = 1$ does not put any upper bound on the norm of B, but from the SVD decomposition $B = U\Sigma V^T$ we have $|\det(B)| = \det(\Sigma)$ so the largest singular value is bigger than unity hence $\|B\|_2 \geq 1$. By restricting B to be in SL(n), we identify all matrices of the form αB for $\alpha \in \mathbb{R} - \{0\}$ with B. Thus we hope we can avoid converging to the trivial infimum of J_1 at $B = 0$.

To find the gradient flow of J_1 over SL(n) we first state this useful lemma, about finding the gradient of a function on a sub-manifold from its gradient on the manifold [Helmke]:

Lemma 6.1 *: Let $f : M \rightarrow \mathbb{R}$ be a smooth function on a Riemannian manifold M and let $V \subset M$ be a sub-manifold, endowed with the Riemannian metric induced from M. If $x \in V$, then the gradient of the restriction $f|_V : V \rightarrow \mathbb{R}$ is the orthogonal projection of $\nabla f(x) \in T_x M$ on the tangent space of V at x, i.e. $T_x V$.*

We state a simple lemma that gives the orthogonal projection of a any matrix on the linear space of matrices of zero trace, namely $\mathfrak{sl}(n)$.

Lemma 6.2 *: The orthogonal projection of the matrix $A_{n\times n}$ on the space of matrices with zero trace is given by: $A^0 = A - \frac{tr(A)}{n} I_{n\times n}$.*

Proof: Obviously $tr(A^0) = 0$ and $tr(A^{0T}(A - A^{0T})) = tr(A)tr(A^{0T})/n = 0$. Therefore A^0 is the orthogonal projection of A to the space of zero trace matrices.$\triangle$

Now we are ready to give the gradient flow of $J_1(B)$ on SL(n). The next two theorems give the Natural gradient of J_1 on SL(n).

Theorem 6.2 *: The Natural gradient of $J_1(B)$ on SL(n) is:*

$$\nabla J_1(B) = 4\Delta^0 B \tag{6.5}$$

where Δ^0 *is the orthogonal projection of*

$$\Delta = \sum_{i=1}^{N} \Big(BC_iB^T - \mathrm{diag}\big(BC_iB^T\big) \Big) BC_iB^T$$

on $\mathfrak{sl}(n)$ *from the above lemma. Furthermore the natural gradient flow to minimize* $J_1(B)$ *on SL(n) is:*

$$\dot{B} = -\nabla J_1(B) = -\Delta^0 B \tag{6.6}$$

with $B(0) \in$ *SL(n).*

Proof: Just apply Lemma's 6.1 and 6.2.

For the convergence properties of this flow, we resort to the Theorem 5.2. As a result we can say that if the solution to (6.6) stays on SL(n) then it converges to a single critical point (more likely to a local minimum) of J_1 on SL(n).

6.3 Nonholonomic Flow for Joint Diagonalization

A more general way to deal with non-compactness of GL(n) and scale-variability of $J_1(B)$ is to project the gradient onto an appropriate space in order to restrict the reduction in the cost function to only desired directions. For instance we may be able to prevent undesirable reduction in the cost function due to diagonal matrices. In other words we want to constrain the flow such that it does not reduce the cost function due to row scaling. For this purpose we introduce the concept of group actions and orbits.

Remark: Nonholonomic constraints are constraints on a vector field or velocity that are not integrable. Nonholonomic treatment of the ICA problem is related

to the issue of scale indeterminacy in the un-mixing matrix and is a well known property (see for example Chapters 2 and 3 in [Haykin] as well as [Amari 2]).

6.3.1 Group Action on a Manifold

Definition 6.1 *: A (left) action of a Lie group G with identity element e on a manifold M is a smooth mapping $\Phi : G \times M \to M$ such that :*

1. $\Phi(e, x) = x$ *for all* $x \in M$,
2. $\Phi(g, \Phi(h, x)) = \Phi(gh, x)$ *for all* $g, h \in G$ *and* $x \in M$.

For any $x \in M$ the orbit of x is defined as $\mathcal{O}_x = \{\Phi(g, x) | g \in G\} \subset M$

Example: Let G be GL(n) and M the space of $n \times n$ matrices. Then the action defined by: Sim : GL(n) $\times M \to M$ with $(S, X) \mapsto SXS^{-1}$ is the similarity transformation which preserves the set of eigenvalues of X. The orbit of this action is $\mathcal{O}_X = \{SXS^{-1} | S \in \text{GL}(n)\}$, that is all the matrices that have the same eigenvalues as X.

It is easy to see that for any action the relationship "$x \sim y, x, y \in M$ iff $y \in \mathcal{O}_x$"is an equivalent relationship on M and the orbits are its equivalent classes. In the above example the relationship of "having the same set of eigenvalues" is the corresponding equivalence relationship.

6.3.2 The Action of the Group of Diagonal Matrices

Consider the group of nonsingular diagonal matrices $\mathcal{D}$. Let's define an action $\Phi : \mathcal{D} \times \text{GL}(n) \to \text{GL}(n)$ by $(D, B) \mapsto DB$, i.e. left multiplication by a non-singular diagonal matrix. The orbit of a matrix B is the set $\mathcal{O}_B = \{DB | D \in \mathcal{D}\}$. We can show that $\mathcal{O}_B$ is an n dimensional sub-manifold of GL(n). The tangent

space to this sub-manifold at B is $T_B^{\mathcal{O}_B} = \{\Delta B | \Delta_{n\times n} \text{is diagonal}\}$. On the other hand the tangent space to GL(n) at B is $T_B^{GL(n)} = \{\Delta B | \Delta \in \mathbb{R}^{n\times n}\}$. Thus $T_B^{\mathcal{O}_B} \subset T_B^{GL(n)}$. The orthogonal complement of the of $T_B^{\mathcal{O}_B}$ in $T_B^{GL(n)}$ is given the next proposition.

Lemma 6.3 *: At any point $B \in GL(n)$ the orthogonal complement with respect to the Natural Riemannian metric of $T_B^{\mathcal{O}_B}$ in $T_B^{GL(n)}$ is the set $T_B^{\mathcal{O}_B^{\perp}} = \{\Delta B | \text{diag}(\Delta_{n\times n}) = 0\}$.*

Proof: $T_B^{\mathcal{O}_B}$ is an n dimensional linear subspace of $T_B^{GL(n)}$ with bases $\{\Delta_i B\}_{i=1}^n$ where Δ_i is a diagonal matrix whose only non zero diagonal element is the ii^{th} entry and is equal to unity. On the other hand $T_B^{\mathcal{O}_B^{\perp}}$ is also a linear $n^2 - n$ dimensional subspace of $T_B^{GL(n)}$ with bases $\{\Delta_{ij}^{\perp} B\}_{i,j=1}^n$ for $i \neq j$ where $\Delta_{ij}^{\perp}$ is an $n \times n$ matrix whose only non zero element is at the ij^{th} position and is unity. For any k and $i \neq j$ between 1 and n let $\xi = \Delta_k B$ and $\eta = \Delta_{ij}^{\perp} B$, then we have:

$$\text{tr}((\xi B^{-1})^T (\eta B^{-1})) = \text{tr}(\Delta_k^T \Delta_{ij}^{\perp}) = 0$$

Hence every vector in $T_B^{\mathcal{O}_B^{\perp}}$ is perpendicular to all vectors in $T_B^{\mathcal{O}_B}$, and because the sum of their dimensions is n^2, they are orthogonal complement with respect to the Natural Riemannian inner product.$\triangle$

Using this lemma we can project any tangent vector to GL(n) at B to two orthogonal components, one along $T_B^{\mathcal{O}_B}$ and the other orthogonal to that. The direction along $T_B^{\mathcal{O}_B}$ amounts to the direction in which the cost function is reduced by only diagonal matrices. Therefore by restricting the gradient to $T_B^{\mathcal{O}_B^{\perp}}$ we can avoid the contribution of diagonal matrices in cost reduction. Let's define for any matrix Δ:

$$\Delta^{\perp} = \Delta - \text{diag}(\Delta) \tag{6.7}$$

We have this theorem, afterwards:

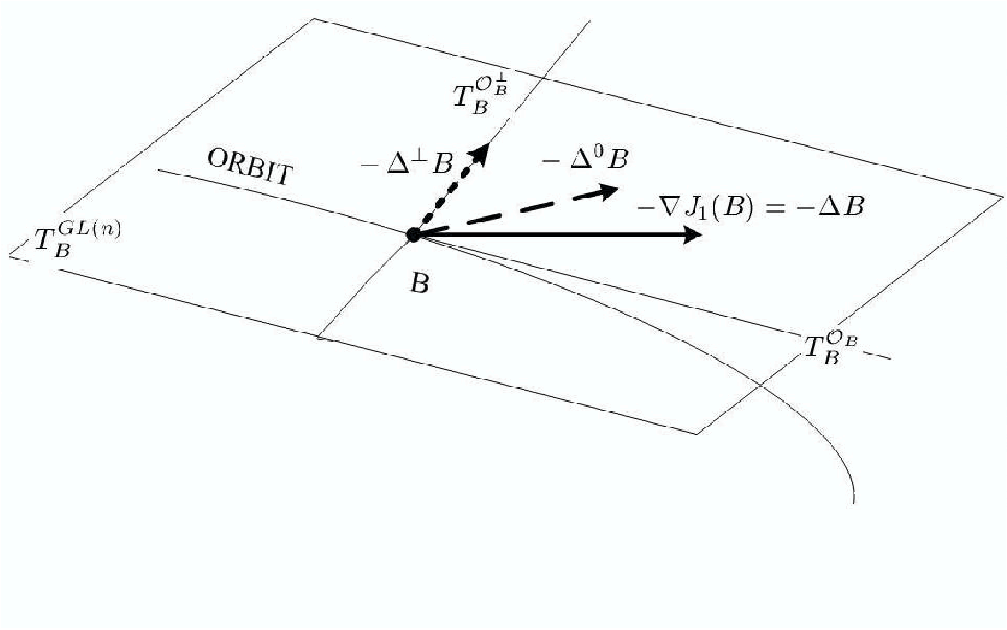

Figure 6.1: Vector fields in equations (6.4),(6.6) and (6.8)

Theorem 6.3 *: The Natural gradient flow for minimization of* $J_1(B)$ *restricted to* $T_B^{\mathcal{O}_B^\perp}$ *is given by:*

$$\dot{B} = -\Delta^\perp B \tag{6.8}$$

where $\Delta = \sum_{i=1}^{N} \big(BC_iB^T - \mathrm{diag}\big(BC_iB^T\big)\big)BC_iB^T$.

Proof: It is the immediate result of theorem (2) and Lemma (3).$\triangle$

Figure (6.1) illustrates how the Natural gradient flow on SL(n) (6.6) and the non-holonomic flow in (6.8) are related to the original GL(n) Natural gradient flow of $J_1(B)$.

It is interesting to verify whether the projected flow in (6.8) still gives a descent flow or not. That is, is the time derivative of $J_1(B)$ negative? We can see from the definition of gradient with respect to the Natural Riemannian:

$$\dot{J}_1 = \mathrm{tr}((\nabla J_1 B^{-1})^T \dot{B} B^{-1}) = -\mathrm{tr}(\Delta^T \Delta^\perp) = -\mathrm{tr}(\Delta^{\perp T} \Delta^\perp) = -\sum_{i \neq j} \Delta_{ij}^2 \leq 0$$

So as long as Δ is not diagonal, (6.8) is a descent flow which is a desirable property. Note that if the initial condition $B(0) \in \mathrm{SL}(n)$, then flow (6.8) can be also considered as a flow over $\mathrm{SL}(n)$, hence $\det(B) = 1$ and $\|B\|_2 \geq 1$.

6.4 Flows on the Manifolds of Triangular Matrices

It is easy to see that the groups of $n \times n$ non-singular lower LL(n) and non-singular upper triangular matrices UL(n) are Lie groups. The sub-group of lower triangular matrices with all diagonal elements equal to unity SLL(n)$\subset$ LL(n) and the sub-group of upper triangular matrices with all diagonal elements equal to unity SUL(n)$\subset$ UL(n) are also Lie groups of dimension $\frac{n(n-1)}{2}$. In fact they are very simple sub-manifolds of GL(n) and SL(n). $\mathfrak{ll}(n)$ is the projection of $\mathfrak{gl}(n)$ over the space of lower triangular matrices and $\mathfrak{sll}(n)$ is the projection of $\mathfrak{gl}(n)$ over the space of lower triangular matrices with zero diagonal. The Lie Algebras of UL(n) and SUL(n) can be found in a similar way. Note that we did not define SUL(n) as upper triangular matrices with unity determinant as we shall find the presented definition more useful.

We can easily find the (upper or lower) triangular counterpart of the derived flows for minimizing J_1. Flows (6.6) and (6.8) are of the form $\dot{B} = -\Delta B$ where Δ is defined accordingly. In fact this is the form of any tangent vector at B. So by projecting Δ to the space of triangular matrices (lower or upper) we can find the triangular (lower or upper) version of the derived flows. Let Δ^L denote a lower triangular matrix that has the same lower triangular part as Δ. Then the lower

triangular version of flow (6.8) is

$$\dot{B} = -\Delta^{\perp L} B, \quad B(0) \in \mathrm{SLL(n)} \tag{6.9}$$

with the same Δ has defined in the Theorem 6.3. The corresponding upper triangular version is derived in the same manner.

The main reason for introducing flows on triangular matrices is that triangular versions of the nonholonomic flow (6.8) are suitable for discretization, in the sense that it is easy to have updates of lower or upper triangular matrices that produce unity determinant sequence of matrices, as we shall see in the next section.

6.5 Discretization of the Flows

As for discretization of flows over $\mathrm{O}(n)$ in Chapter 5, here also we have the problem of keeping the updates on the manifold. For flows introduced so far this translates to having updates with unity determinant. In this section we develop some methods to discretize the flows introduced in the previous sections. Here we first give Euler and explicit fourth order Runge-Kutta discretization of the flows (see Section 5.5). After that we will use the idea of LU decomposition of the un-mixing matrix to derive a method that keeps the updates on the manifold by construction. Finally we incorporate the Armjio line search to Euler discretization.

As we mentioned all the flows developed so far have either the form:

$$\dot{B} = -\Delta B, \quad B(0) = I_{n\times n} \tag{6.10}$$

6.5.1 Euler Discretization

Using the Euler method for (6.10) we have the update:

$$B_{k+1} = (I_{n\times n} - \mu\Delta_k)B_k \quad k \geq 0 \tag{6.11}$$

Algorithm 6.1

1. **Set** μ **and** ϵ.
2. **Set** $B_0 = I_{n\times n}$ or to a good initial guess.
3. **While** $\|\Delta_k\|_F > \epsilon$ **do**

$B_{k+1} = \big(I - \mu\Delta_k\big)B_k$

if $\|B_{k+1}\|_F$ is "*big*" **then** *reduce* μ **and goto** 2.

4. **End**

Table 6.1: A fixed-step size implementation of the Euler discretization of the gradient flow (6.10), which represents all the continuous flows developed based on the Natural Riemannian metric. The unspecified parameters and qualities are to be decided in practice.

where Δ_k is computed according to the corresponding flow at each step and μ is a small enough step-size so that B_{k+1} is on the desired manifold (i.e having unity determinant) and B_k is bounded. Table(6.1) shows Algorithm 6.1 that represents this discretization. Note that we have not provided any measure to check the determinant of B_k, as it is very costly. Only by choosing small μ we hope we can achieve this goal to a good extent. However checking whether B_k is blowing up and the algorithm is diverging is easy. If μ is small enough such that $J_1(B_{k+1}) < J_1(B_k)$ then we have a *descent* algorithm. Therefore for two reasons of keeping the update on the manifold and descent, we need to have small step-size. We emphasize that in an optimization problem on a vector space only the second restriction appears and as long as we have a descent we can choose large step sizes.

6.5.2 Fourth Order Runge-Kutta Discretization

In this section we proceed much the same way as Section 5.5 to derive RK discretization for the developed flows. Considering equation (6.10) we expect that the

update equation for B_k in an RK scheme will be in the form of $B_{k+1} = (I+\mu\Psi_k)B_k$ where the matrix Ψ_k is computed using stage values in the RK method. Based on the ideas used in developing our flows we may want to impose some conditions on Ψ_k such that the discrete scheme more resembles the continuous one. In particular for a nonholonomic flow we can require Ψ_k to have zero diagonal and for a gradient flow on SL(n) we project Ψ_k to Ψ_k^0. Note that Ψ_k will not have these properties immediately contrary to the Δ_k matrix in the Euler method. Table (6.2) shows an algorithm based on explicit fourth order RK method that takes this measure in to account. The underlying flow can be on SL(n), nonholonomic or flow on triangular matrices manifold. Note that for flows over SLL(n) and SUL(n) we do not require to alter Ψ_k, because it will have the required properties by construction.

6.5.3 An iterative algorithm based on LU factorization

Here we introduce an iterative algorithm based on LU [Golub] factorization of the un-mixing matrix B and the nonholonomic flow introduced before. The idea is to have alternating nonholonomic flows over SLL(n) and SUL(n). Consider the nonholonomic flow when $B(t)$ is confined to be in SLL(n), i.e (6.9). Note that both the Euler and RK discretization will have the form $B_{k+1} = (I + \mu_k\Phi_k^{\perp L})B_k$, where $\Phi_k^{\perp L}$ is a matrix whose non-zero elements are below the diagonal. Therefore $\det B_{k+1} = \det B_k$ and if $B_0 = I_{n\times n}$ then $\det B_k = 1$ for all k by construction, moreover all the diagonal elements of B_k are one. The same holds if B is confined to be in SUL(n) and $B_0 = I_{n\times n}$. By combining these two approaches we can have an iterative algorithm that alternatively searches for upper and lower triangular factors of the un-mixing matrix and keeps the determinant unity by construction. An algorithm for this is presented in Table (6.3).

Algorithm 6.2

Consider any one of the flows (6.6), (6.8) or (6.9). Let them be written in the general form $\dot{B} = \Delta_B B$ where Δ_B is found for each of them accordingly.

1. **Set** μ **and** ϵ.

2. **Set** $B_0 = I_{n\times n}$ or to a good initial guess.

3. **While** $\|\Delta_k\|_F > \epsilon$ **do**

$Y_1 = B_k$, $Y_2 = (I + \frac{1}{2}\mu\Delta_{Y_1})B_k = \Phi_{Y_1}B_k$

$Y_3 = (I + \frac{1}{2}\mu\Delta_{Y_2}\Phi_{Y_1})B_k = \Phi_{Y_2}B_k$, $Y_4 = (I + \mu\Delta_{Y_3}\Phi_{Y_2})B_k = \Phi_{Y_3}B_k$

$\Psi_k = \frac{1}{6}\Delta_{Y_1} + \frac{1}{3}\Delta_{Y_2}\Phi_{Y_1} + \frac{1}{3}\Delta_{Y_3}\Phi_{Y_2} + \frac{1}{6}\Delta_{Y_4}\Phi_{Y_3}$

If discretizing (6.6) then $\Psi_k \leftarrow \Psi_k^0$

If discretizing (6.8) then $\Psi_k \leftarrow \Psi_k^{\perp}$

$B_{k+1} = (I + \mu\Psi_k)B_k$

if $\|B_{k+1}\|_F$ is "*big*" **then** *reduce* μ **and goto** 2.

4. **End**

Table 6.2: A fixed-step size implementation of the fourth order RK discretization of the flow (6.10), which can represent all the continuous flows developed. The unspecified parameters and qualities are to be decided in practice.

The most interesting feature of this algorithm is that $\det(L_k) = \det(U_k) = 1$ for all k independent of μ_k and hence $\det(B) = 1$ by construction. This is a nice property, because it guarantees that B is non-singular and $\|B\|_2 \geq 1$.

6.5.4 Incorporation of the Armijo line search method

The Armijo line search method is a well known line search method to find step size for descent methods [Tits]. We can incorporate this method in the above algorithm (when the Euler method is used for discretization), knowing the fact that using

Algorithm 6.3

Consider the set $\{C_i\}_{i=1}^N$ of symmetric matrices. Let (a):$\dot{U} = -\Delta^{\perp U} U$ and (b): $\dot{L} = -\Delta^{\perp L} L$ with $U(0) = L(0) = I$ be the corresponding upper and lower nonholonomic joint diagonalization flows. (See equations (6.8) and (6.9))

1. Use Algorithm 6.1 or 6.2 to find U the solution to (a).
2. **Set** $C_i \leftarrow U C_i U^T$.
3. Use Algorithm 5.1 or 5.2 to find L the solution to (b)
4. **Set** $C_i \leftarrow L C_i L^T$.
5. **Set** $B \leftarrow L\ UB$
6. **If** $\|LU - I\|_F$ is "small" stop **else goto** 1

Table 6.3: The pseudo code for an algorithm for joint diagonalization of $\{C_i\}_{i=1}^N$, based on the LU factorization of the un-mixing matrix. The nonholonomic flows corresponding to L and U are discretized using Algorithm 5.1 or 5.2. The algorithm alternatively finds U and L and forms B from them sequentially. The main feature of this algorithm is that $\det B = 1$. This is a nice property, because it guarantees that B is non-singular and $\|B\|_2 \geq 1$. The unspecified parameters and qualities are to be decided in practice.

the above algorithm the updates are always on SL(n) independent of step-size. The basic idea in the Armjio method is to change the step-size $\mu_k = \beta^j$ where $\beta \in (0, 1)$ by changing j and keeping β constant to find the smallest j such that $J_1(B_{k+1}) - J_1(B_k) \leq \alpha \beta^j \dot{J}_1(B_k)$ where $\alpha \in (0, 1)$ and B_{k+1} is updated according to any descent direction. Note that $\dot{J}_1(B_k)$ is the directional derivative in the descent direction. The derived algorithm is coded in Table (6.4).

Algorithm 6.4

Consider the set $\{C_i\}_{i=1}^N$ of symmetric matrices.

1. **Set** $B = I_{n\times n}$, **set** $\alpha \in (0,1)$.

2. $U_1 = I_{n\times n}$ **do until** "convergence":

find the smallest nonnegative integer j

such that $J_1(U_{k+1}) - J_1(U_k) \leq \alpha\beta^j tr(\Delta_k^{\perp UT}\Delta_k^{\perp U})$ where: $U_{k+1} = (I - \beta^j\Delta_k^{\perp U})U_k$

and $\Delta_k = \sum_i \big(U_k\, C_i\, U_k^T - \mathrm{diag}\big(U_k\, C_i\, U_k^T\big)\big)U_k\, C_i U_k^T$ and increment k.

3. Let U be the result of the previous step, **set** $C_i \leftarrow UC_iU^T$.

4. $L_1 = I_{n\times n}$ **do until** "convergence":

find the smallest nonnegative integer j

such that $J_1(L_{k+1}) - J_1(L_k) \leq \alpha\beta^j tr(\Delta_k^{\perp LT}\Delta_k^{\perp L})$ where: $L_{k+1} = (I - \beta^j\Delta_k^{\perp L})L_k$

and $\Delta_k = \sum_i \big(L_k\, C_i\, L_k^T - \mathrm{diag}\big(L_k\, C_i\, L_k^T\big)\big)L_k\, C_i L_k^T$ and increment k.

5. Let L be the result of previous step, **set** $C_i \leftarrow LC_iL^T$.

6. **Set** $B \leftarrow LUB$.

7. **If** $\|LU - I\|_F$ is "small" stop **else goto** 2

Table 6.4: The pseudo code for an algorithm for joint diagonalization of $\{C_i\}_{i=1}^N$, based on the LU factorization of the un-mixing matrix and Armijo line search. The nonholonomic flows corresponding to L and U are discretized using the Euler method. By construction $\det(U_k) = \det(L_k) = 1$, so the Armijo method can be used without any restriction on μ_k for keeping updates on SL(n). The algorithm alternatively finds U and L and forms B from them sequentially. The main feature of this algorithm is that $\det B = 1$. This is a nice property, because it guarantees that B is non-singular and $\|B\|_2 \geq 1$. The unspecified parameters and qualities are to be decided in practice.

Chapter 7

ICA/BSS Algorithms Based on Joint Diagonalization

In this chapter we develop ICA/BSS algorithms based on joint diagonalization methods developed before. We introduce hybrid methods that whiten the data, however do not restrict the subsequent search space to orthogonal matrices.

7.1 Introduction

In Chapter 5 we gave a gradient based version of the JADE algorithm for BSS/ICA. In this chapter we will develop more general gradient based algorithms for BSS/ICA using methods developed in the previous chapters. These algorithms do not restrict the search space to orthogonal matrices. That is, they search for non-singular joint diagonalizer for a set of cumulant matrix slices.

Consider the ICA model:

$$\vec{\mathbf{x}} = A\vec{\mathbf{s}} + \vec{\mathbf{n}} = \vec{\mathbf{z}} + \vec{\mathbf{n}} \tag{7.1}$$

with standard assumptions, especially with $\vec{\mathbf{n}}$ being Gaussian noise. In lack of

information about noise or $\vec{\mathbf{z}}$'s covariance matrix we choose to whiten the data based on the (estimated) covariance matrix of $\vec{\mathbf{x}}$. Suppose W is a matrix that whitens or shperes $\vec{\mathbf{x}}$, then for the white signal $\vec{\mathbf{y}}$ we have:

$$\vec{\mathbf{y}} = W\vec{\mathbf{x}} = WA\vec{\mathbf{s}} + W\vec{\mathbf{n}} = A_1\vec{\mathbf{s}} + \vec{\mathbf{n_1}} \tag{7.2}$$

This can be considered as a new problem with the same structure as before where $A_1 = WA$ is non-singular and $\vec{\mathbf{n_1}}$ is again a Gaussian noise. Obviously the form of the problem has not changed and we have not lost any information. Yet this new data is closer to independence "in most cases" (see Section 2.4.4). This is the traditional uncorrelating step used in different contexts, but obviously we shall proceed further!.

Note that when the noise is not very strong we expect the reduced un-mixing matrix namely $A_1 = WA$ to be an almost orthogonal matrix. This is an important fact because in practice it makes the subsequent nonorthogonal joint diagonalization much easier. We elaborate more on this. As a special case consider the case where the covariance matrix of $\vec{\mathbf{n}}$ in (7.1) has the form $R_{\mathbf{nn}} = \sigma^2 I_{n\times n}$. Then it is not difficult to show that (with the convention that $R_{\mathbf{ss}} = I_{n\times n}$) for the 2-norm of orthogonality error we have that (in the first order of approximation):

$$\|A_1 A_1^T - I_{n\times n}\|_2 \leq \alpha(\frac{\sigma}{\nu_{min}})^2 \tag{7.3}$$

where ν_{min} is the smallest singular value of A and α is a positive constant. This means that if the noise power is not strong compared to the smallest singular value of A the matrix $A_1 = WA$ in (7.2) is close to orthogonal. However, if we assume the orthogonality and impose that on the subsequent joint diagonalization phase, obviously we are reducing the degree's freedom in the search and abandon further possible joint diagonalization. Hence we should perform non-orthogonal

joint diagonalization. Note that an orthogonal flow has the form $\dot{\Theta} = \Delta\Theta$ where Δ is skew-symmetric, so we have only $\frac{n(n-1)}{2}$ degrees of freedom in updating Θ whereas the same scheme for a nonholonomic flow can have $n(n-1)$ degrees of freedom. This has the advantage that makes more minimization possible but on the other hand the compactness of $\mathrm{O}(n)$ is lost. As we mentioned before and as actual simulations show the JD methods developed in Chapter 6 work much better in the case the sought matrix is close to orthogonal. To motivate this we recall that if the initial condition for in the JD algorithm is the identity and the sought matrix is close to orthogonal the required change in the norm is not significant. Note that this can be improved if we initialize the JD algorithm with a better guess, which can be for example the eigen matrix for one of cumulant slices. However we will not choose this method in our simulations.

Considering (7.2) and using the properties of cumulants and cumulant slices we have that $Cum_{\mathbf{y}}(:,:,i,j) = A_1\Lambda_{ij}A_1^T$ where Λ_{ij} is a diagonal matrix from Lemma 2.1 (see also equation (3.2)). Now we can search for a non-singular matrix B that jointly diagonalizes $\{Cum_{\mathbf{y}}(:,:,i,j)\}_{i,j}$ using methods developed in Chapter 6 for non-orthogonal JD. To be rigorous we mention that this deduction was based on the assumption that the exact correlations and cumulants of the data are available. Mainly the assumptions that $Cum_{\mathbf{n_1}}(:,:,i,j) = 0$ and that $Cum_{\mathbf{y}}(:,:,i,j)$ are diagonalizable by A_1 are based on using exact cumulants. Certainly deviation from these assumptions in small sample sizes will affect the performance of the algorithms.

There are other benefits to whitening the data, for example the whitened signal usually is such that its cumulant matrix slices have the same dynamic range and again this helps the numerical stability. We should also mention that covariance is

the most reliable statistics (compared to higher order statistics), in the sense that it has the least amount error estimation. Therefore it is convincing to use as much information as we can from covariance. We mention that there are methods that rely only cumulants (see for example [De Lathauwer2]), in this view these methods are missing the nice properties of second order statistics.

In summary, in order to do ICA, we try to reduce the mutual information "globally or coarsely" at the first step by whitening the data and then by joint diagonalization of cumulant slices we reduce the mutual information further "locally or finely". This way we use the benefits of *white data*, higher order statistics and the group structure of both *orthogonal* and *non-singular* matrices.

7.2 A Class of ICA Algorithms Based on Non-Orthogonal JD

Here we introduce a general scheme for our algorithms. Assume that T realizations of data in (7.1) with standard assumptions are available. We place these realizations column-wise in an $n \times T$ matrix X_T. So we can also write $X_T = AS_T + N_T$. The Whitening step WHT consists of computing $R_{XX} = \frac{1}{T} X X^T$ and then $W = R_{XX}^{-\frac{1}{2}}$ to find $\vec{\mathbf{y}} = W\vec{\mathbf{x}}$ or $Y_T = WX_T$. In practice we always use vectors after we have made their mean zero. The cumulant computing step CUM is simply computing the fourth order cumulants of Y_T using sample estimates. We can take any subset of the computed cumulant slices. The number of cumulant slices is n^2, so for large n using all slices maybe prohibitive. Let $\{C_i\}_{i=1}^{N}$ be the subset chosen.

To proceed further we adopt the idea of sequentially minimizing the cost function:

$$J_1(B, \{C_i\}_{i=1}^N) = \sum_{i=1}^{N} \left\| BC_iB^T - \mathrm{diag}(BC_iB^T) \right\|_F^2$$

in a multiplicative fashion. That is, first we find an orthogonal Θ to minimize J_1 (we call this step JDO) and we recompute $C_i \leftarrow \Theta C_i \Theta^T$ (we call this step RCO) and then again minimize J_1 (this time with a new $\{C_i\}_{i=1}^N$, of course) by a non-orthogonal matrix B_{JDN} (step JDN). Thereafter $B \leftarrow B_{JDN}\Theta B$. We can proceed further and again recompute $C_i \leftarrow B_{JDN}C_iB_{JDN}^T$ (step RCN) and repeat the previous steps. The reason for orthogonal joint diagonalization is two-fold. First as it was mentioned, after whitening we expect the un-mixing matrix to be close to an orthogonal one; second, because of compactness of O(n) and the theoretical assurance of convergence of the gradient flows on O(n) (presented in Section 5.2) we may want to perform orthogonal JD. Note that the SL(n) flow developed in Section 6.2 is a gradient flow on a non-compact set, and also the nonholonomic flows developed in Section 6.3 are not flows on a compact set. Notice thatsteps JDO-RCO-JDN together are equivalent to JDN when it is initialized to the initial condition $B_{JDN0} = \Theta$. This is obviously the result of the group structure of the problem. Another point to be mentioned is that for all the algorithms developed in the previous chapters both the initial conditions and answers are matrices with unity determinant, so an orthogonal matrix can be used as the initial condition for them. Of course, despite this we can avoid JDO-RCO and just do JDN with $B_{JDN0} = I_{n\times n}$.

Another approach maybe to update the data after any of the JD procedures and then recompute the cumulants from the data samples, this way flow the data along with the un-mixing matrix. While this approach seems plausible it is much more costly and in practice does not offer performance increase, as it was experienced.

Table (1) summarizes the steps mentioned above and possible algorithms that can be used in each of them. Note that in step RCO we have included the original Jacobi based Algorithm 3.1 for orthogonal JD, and also in both RCO and JDN we can use any other ODE solver, such as the popular $MATLAB$®'s "ode45" routine.

WHT: Whiten the data, let $B = W$ be the whitening matrix and $\vec{\mathbf{y}}$ the whitened data.
CUM: Compute $C = \{C_i\}_{i=1}^{N}$ a subset of forth order Cumulant matrix slices for $\vec{\mathbf{y}}$.
JDO: Jointly Diagonalize C by an Orthogonal matrix using any of Algorithms 3.1, 5.1, 5.2 or any other ODE solver.
RCO: Re-Compute $C_i \leftarrow \Theta C_i \Theta^T$ and $B \leftarrow \Theta B$, where Θ is the (Orthogonal) answer to step JDO.
JDN: Jointly Diagonalize C by a Non-orthogonal matrix using any of Algorithms 6.1,2,3,4 or any other ODE solver.
RCN: Re-Compute $C_i \leftarrow B_{JDN} C_i B_{JDN}^T$ and $B \leftarrow B_{JDN} B$, where B_{JDN} is the (Non-orthogonal) answer to step JDN.

Table 7.1: The core steps used in ICA/BSS algorithms for model (7.1).

The usual JADE, EG-JADE or RKG-JADE algorithms can be coded as:

1. **do** WHT
2. **do** CUM
3. **do** JDO **and** RCO
4. If not satisfactory **go to** step 3

We can generalize this to a class of algorithms in the form:

1. **do** WHT
2. **do** CUM
3. **do** (JDO and RCO) **and do** (JDN and RCN)
3'. **do** (JDN and RCN)
4. **If** "not-satisfactory" **goto** 3 or 3' **else END**

Table 7.2: The general scheme for a class of ICA/BSS algorithms based on the JD criterion.

Note that step 3 has two versions. In the first version an orthogonal and a unity determinant matrix are sought and in the second version only a unity determinant matrix is found. As we mentioned they can be equivalent if we initialize JDN with the answer of JDO. In practice we refrain from repeating steps 3 or 3' because it is unlikely to be fruitful unless the stop criteria chosen is large so that major further improvement was possible.

7.3 Numerical examples

In this section we shall consider applying the developed ICA/BSS algorithms to some BSS/ICA problems. As it is evident we have developed a class of algorithms that can have lots of members. For implementation of each step we can have different algorithms with different parameters (for example μ and ϵ in all discretized gradient or the number of iterations in Algorithm 6.3). Hence comparing all these methods will be tedious and even not very useful, because these algorithms are to work with stochastic data, hence their performance also depends on the source probability distribution as well. Therefore we try to give few examples that cap-

ture different aspect of the algorithms.

Example 1: Let us consider $\vec{\mathbf{x}} = A\vec{\mathbf{s}} + \sigma\vec{\mathbf{n}}$ where $\vec{\mathbf{s}}$ is a $n = 5$ dimensional random vector for which the first element is exponentially distributed by parameter $\lambda = 1$ and mean one, the second one has the two sided exponential distribution and the last three are uniformly distributed in $[-\frac{1}{2}, \frac{1}{2}]$, and $\vec{\mathbf{n}}$ is a standard Gaussian noise and σ^2 indicates the actual power of noise at each sensor. The mixing matrix is randomly generated as:

$$A = \begin{bmatrix} -0.42438571607 & -1.03974356765 & 0.85444609063 & 0.20440694544 & -1.42107846008 \\ -0.9210520223 & -0.00242328109 & -1.40278316492 & -0.54986431848 & -0.12122193071 \\ 1.01156524988 & 0.20298091664 & -0.12445806045 & 0.39315798257 & 0.56608725333 \\ -0.35265942639 & 0.74153122094 & 0.43137720844 & -0.84647982436 & -1.62052834709 \\ -0.27103430011 & 0.05900889251 & 0.59195584499 & 1.29681290480 & 0.17889884891 \end{bmatrix}$$

We generate $T = 2000$ independent samples of $\vec{\mathbf{x}}$ with $\sigma = 0$. For the non-orthogonal JD step in this example we only consider nonholonomic flows (equation (6.8) and not gradient flow on SL(n) (equation (6.6). In the first experiment we apply steps 1,2,3 from Table (2) using JDO and JDN by the Euler method (Algorithms 5.1 and 6.1, respectively) with $\mu = .01$ and $\epsilon = .01$. Next we do the same by using fourth order RK method (Algorithms 5.2 and 6.2) with the same values of μ and ϵ. After that we use the LU based methods with both Euler and RK methods. We set $\epsilon = .0001$ and use the same μ to replace the Euler JDN by an Euler LU JDN step. Note that decreasing ϵ is necessary now because the matrix that its norm is considered for stop criteria is a triangular matrix. We repeat the LU steps 5 times. Next we do step 3' only, that is we only do JDN and we forget about an orthogonal diagonalizer. Table (7.3) shows the result of these experiments. We have computed the performance index $Index(P)$ for $P = B\Theta WA$(see equation (2.6)). Note that when we do only JDN then $\Theta = I_{n\times n}$ in computing P.

We also computed det B and $\|\Theta\Theta^T - I\|_F$ as measures of staying on the manifolds. Note that all methods perform almost equally. Of course this can not be a exact way of comparing them. Next we repeat the same experiments by adding noise to

	$Index(P)$	$\det(B)$	$\|\Theta\Theta^T - I\|_F$
Euler JDO/JDN	1.0319	0.9998	0.0397
RK JDO/JDN	1.0326	0.9999	0.0235
Euler JDO/LU-JDN	1.0376	1	0.0397
RK JDO/LU-JDN	1.1262	1	0.0235
Euler JDN	1.081	1.0065	-
RK JDN	1.1159	1.0072	-

Table 7.3: This table gives the performance measures from applying ICA/BSS methods schemed in Tables (1) and (2) to Example 1, when $\sigma = 0$. The parameters for each method are specified in Example 1. All the discretized flows are nonholonomic ones.

$\vec{\mathbf{x}}$ with $\sigma = .1$. Table (7.4) shows the results for the same parameters as before. Obviously $Index(P)$ has increased by noise. Another point is that the required time to perform the computation is more, which shows that the problem becomes more difficult in noise.

Example 2: In this example we again consider the same mixing matrix A as in Example 2 and the same sources. We compare the performance of four different methods in noise: steps (1,2,3) with Euler discretization, steps (1,2,3) with RK discretization steps (1,2,3) but this time using the $MATLAB^{®}$ ODE solver "ode45" which uses a complicated adaptive step size in RK method, and lastly the JADE algorithm. All flows used are nonholonomic flows. We use two values $T = 2000, 5000$ for sample size. We use $\mu = .001$ and $\epsilon = .01$ for all Euler and RK methods. Figure (1) shows their performance. The plots show that the de-

	$Index(P)$	$\det(B)$	$\|\Theta\Theta^T - I\|_F$
Euler JDO/JDN	1.3874	0.9998	0.0389
RK JDO/JDN	1.3848	0.9998	0.0230
Euler JDO/LU-JDN	1.4725	1	.0389
RK JDO/LU-JDN	1.3872	1	0.0230
Euler JDN	1.4140	1.0046	-
RK JDN	1.414	1.0072	-

Table 7.4: This table is the same as the previous one, except that $\sigma = .1$.

veloped algorithms outperform JADE. On the other hand the experiments show that the computation time for these methods is much longer then JADE's. Hence the better performance of the developed algorithms is at the expense of more computation time. Interestingly the fix-step size methods perform on average slightly better than "ode45" but still at the expense of more time. Yet we should say that a $MATLAB^{®}$ implementation in M-files does not give a good assessment of the computation cost of these methods, because the developed algorithms are only loops of addition-multiplication operations, and this can not be implemented via $MATLAB^{®}$ interpreter in an efficient manner. One advantage of the developed JD methods is that they only require additions and multiplications and no division, so they are suitable for fixed point implementation on DSP processors.

Example 3: In this example we compare the performance of nonholonomic flow and gradient flow on SL(n) for ICA. That is we perform steps WHT, CUM, JDN. For the JDN step we use both Euler and RK discretizations of equations (6.6) and (6.8). We apply these algorithms (Algorithm 6.1 and 6.2) to the data produced

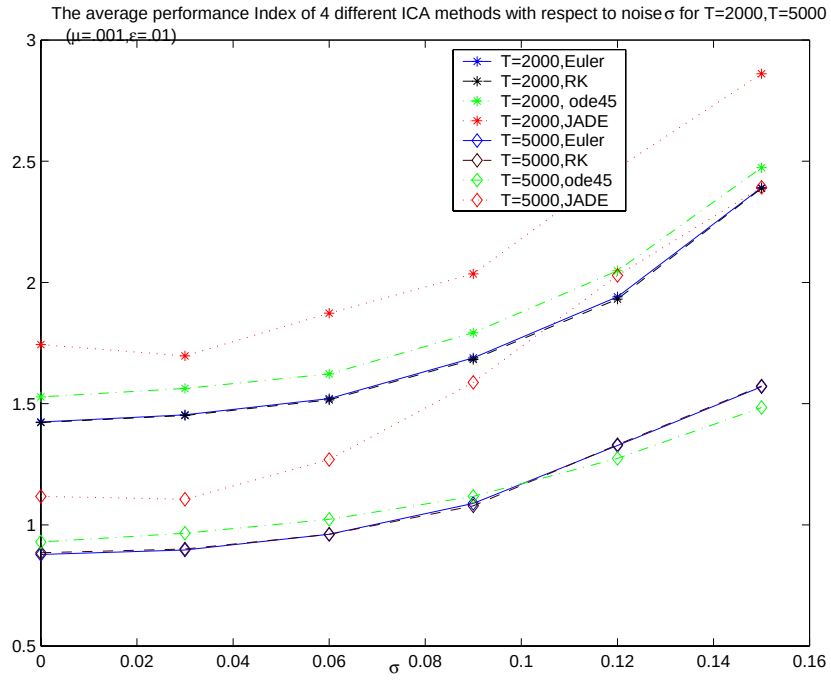

Figure 7.1: (Example 2)This figure shows the average in-noise-performance index (every point is averaged over 100 trials) of algorithms of the form WHT, CUM, JDO-RCO, JDN implemented in three methods: Euler, RK, and "ode45" function in $MATLAB^{®}$. The standard JADE algorithm is also used. These algorithms are applied to the data introduced in Example 1 in the presence of noise. The average $Index(P)$ is plotted versus σ. We consider two sample sizes $T = 2000, 5000$. The parameters for the discretized algorithms are set to $\mu = .001$ and $\epsilon = .01$.

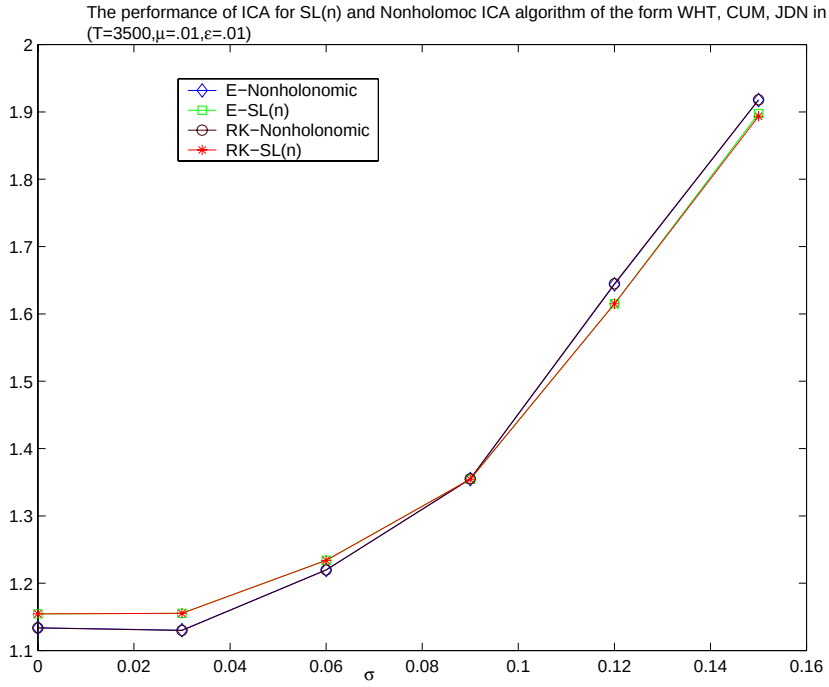

Figure 7.2: (Example 3) This figure shows the average in-noise-performance index (every point is averaged over 100 trials) of algorithms of the form WHT, CUM, JDN based on Nonholonomic flow on GL(n) and gradient flow on SL(n), discretized in both Euler and RK methods. These algorithms are applied to the data introduced in Example 1 in the presence of noise. The average $Index(P)$ is plotted versus σ. We consider two sample sizes $T = 3500$ and all implementation have the same parameters $\mu = .01$ and $\epsilon = .01$. Note that no orthogonal JD (JDO) is applied.

in the same way as in Example (1). We set $\mu = \epsilon = .01$ for all methods and we use $T = 3500$ samples. Figure (7.2) shows the average performance index with respect to noise σ. From the figure it is understood that SL(n) and nonholonomic flows perform almost the same in this problem.

Chapter 8

Summary and Suggestions for Future Work

In summary, after introducing necessary tools and concepts we developed orthogonal and non-orthogonal flows for joint diagonalization of a set of symmetric matrices. This derivation was based on defining a suitable Riemannian metric that matches the group structure of the underlying Lie group (i.e. $\mathrm{O}(n)$ and $\mathrm{GL}(n)$). In the case of orthogonal JD, the cost function $J_1(\Theta)$ in equation (3.3) is minimized with respect to $\Theta \in \mathrm{O}(n)$ resulting in the flow in equation (5.4). Due to compactness of the group $\mathrm{O}(n)$ this cost function has a minimum on $\mathrm{O}(n)$ and hence the gradient minimization of it is a legitimate process whereas in the case of non-orthogonal JD the cost function $J_1(B)$ with $B \in \mathrm{GL}(n)$ does not have this property. $J_1(B)$ can be reduced by diagonal B with small norm and it is not acceptable in the context of ICA/BSS. On the other hand as we showed in Chapter 6 unless all the symmetric matrices to be diagonalized have an exact common diagonalizer $J_1(B)$ has no minimum on $\mathrm{GL}(n)$ and its only minimizer is the trivial $B = 0$ which of course is not in $\mathrm{GL}(n)$. To combat this (or to achieve some sort

of compactification of GL(n)), we followed the idea of identifying B and its multiples to keep $\|B\|_F$ large enough so that the minimization becomes non-trivial. If we identify $\alpha B, \alpha \in \mathbb{R} - \{0\}$ with B then we will have the gradient flow (6.6), which is a flow on SL(n). If we identify ΛB where Λ is a non-singular diagonal matrix we will have flow (6.8) which is a nonholonomic flow with respect to the Natural Riemannian metric on GL(n). These two flows were derived by projecting (restricting) the gradient of $J_1(B)$ on (to) appropriate spaces. Instead of this projection we could use another cost function introduced in equation (3.7), and this can be one idea for further work.

We also gave local point convergence proofs for the orthogonal and SL(n) gradient flows. In the case of the nonholonomic flow still there is room for investigating convergence properties of the flows.

Proper discretization of flows on a manifold is not a trivial task. There are sophisticated methods to do this, we did not follow this path, instead we used Euler and fourth order Runge-Kutta methods with suitable modification and small enough step-size to have discretization schemes that stay on the manifold to a good extend. However, we gave an iterative discretization scheme for flow (6.8) based on the LU decomposition of B. This method has the property that for the resultant B, $\det B = 1$ regardless of the step-size used. We also did not seek adaptive step-size selection schemes in discretization. This can be a possible, but yet little bit dubious path for further work. These gradient based algorithms are sensitive to scaling mismatches in data. Proper pre-conditioning can be very useful in the general case, for as we mentioned in the case of the developed ICA/BSS algorithms the matrices to be diagonalized (i.e. cumulant slices of a white signal) are in general properly scaled.

The JD algorithms based on gradient are very slow, as other gradient methods are, specially if the level curves of the cost functions are not nice, and this is the case when the matrices are far from having a joint diagonalizer. Therefore a very major development would be to devise Newton based methods for this optimization problem, that take the group structure and the issue of non-compactness of $\mathrm{GL}(n)$ in to account. For the case of orthogonal JD the so-called JADE algorithm (Algorithm 3.1) [Cardoso1] is a simple and fast solution.

Based on the developed flows we proposed a class of ICA/BSS algorithms that whiten the data first but do not confine the search for an un-mixing matrix to $\mathrm{O}(n)$ as the JADE algorithm does. In a more elegant language we can say that the developed algorithms solve the ICA problem in two steps: first by a whitening step we get closer to an independent answer (*globally*) and then by looking for a non-orthogonal joint diagonalizer for the fourth order cumulant slices of the whitened signal we *locally* solve the ICA problem. The celebrated JADE algorithm does the same except that the sought matrix is confined to be orthogonal. Our methods are also different from only-HOS based methods because we use the second order statistics to whiten the data. Our argument for doing this is that in general a signal becomes closer to independence by whitening (see Section 2.4.4 for more details). We neither gave a precise statement of this fact nor a proof, which can be a subject for further work as well. It should be noted that by whitening the data we will not lose any information due to the group structure of the ICA/BSS problem. Moreover whitening has the advantage that scales the data properly for further processing (a concept comparable to pre-conditioning in numerical analysis). This is also a vague statement that can be made more rigorous and investigated further.

We applied the developed methods to few cases of synthetic data, a more general

pace would be to examine the performance of these core ICA/BSS methods in more sophisticated and specific-purpose ICA/BSS schemes. The non-orthogonal JD algorithms could also be examined in the context of non-stationary BSS where a set of correlation matrices are to be jointly diagonalized [Pham1].

Appendix A

Derivations and Some Proofs

In some cases we will use these easy-to-prove identities about $n \times n$ matrices A, B, C:

$$\operatorname{tr}(ABC) = \operatorname{tr}(CAB) = \operatorname{tr}(BCA) \tag{A.1}$$

$$\operatorname{tr}(A \operatorname{diag}(B)) = \operatorname{tr}(\operatorname{diag}(A)\ B) = \operatorname{tr}(\operatorname{diag}(A) \operatorname{diag}(B)) \tag{A.2}$$

and if A and B are symmetric the Lie Bracket is skew-symmetric:

$$[A, B]^T = (AB - BA)^T = BA - AB = -[A, B] \tag{A.3}$$

A.1 Proof of Theorem 5.1

We name each single term in the summation as $J_i(\Theta)$ that is:

$$J_1(\Theta) = \sum_{i=1}^{N} J_i = \sum_{i=1}^{N} \left\| \Theta C_{i0} \Theta^T - \operatorname{diag}(\Theta C_{i0} \Theta^T) \right\|_F^2 \tag{A.4}$$

then note that using the above identities and the symmetry of C_{i0} and $\Theta C_{i0} \Theta^T$ and orthogonality of Θ we have:

$$J_i = \operatorname{tr}\Big(\big(\Theta C_{i0} \Theta^T - \operatorname{diag}(\Theta C_{i0} \Theta^T)\big) \big(\Theta C_{i0} \Theta^T - \operatorname{diag}(\Theta C_{i0} \Theta^T)\big) \Big) =$$

$$\mathrm{tr}(C_{i0}^2) - 2\mathrm{tr}\big(\Theta C_{i0}\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) + \mathrm{tr}\big(\mathrm{diag}(\Theta C_{i0}\Theta^T)\ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) =$$

$$\mathrm{tr}(C_{i0}^2) - \mathrm{tr}\big(\Theta C_{i0}\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) \tag{A.5}$$

In the next step we compute the time derivative of J_i, i.e. $\dot{J}_i$. By using (A.1,A.2) we have:

$$\dot{J}_i = -2\mathrm{tr}\big(\dot{\Theta} C_{i0}\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) - 2\mathrm{tr}\big(\Theta C_{i0}\dot{\Theta}^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) \tag{A.6}$$

As it was mentioned in Chapter 4 we can write $\dot{\Theta} = \Theta\Delta$ where $\Delta_{n\times n}$ is a skew-symmetric matrix. With this in mind we continue (A.6) as:

$$\dot{J}_i = -2\mathrm{tr}\big(\Theta\Delta C_{i0}\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) - 2\mathrm{tr}\big(\Theta C_{i0}\Delta^T\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) = \tag{A.7}$$

$$-2\mathrm{tr}\big(\Theta\Delta C_{i0}\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big) + 2\mathrm{tr}\big(\Theta C_{i0}\Delta\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\big)$$

Then using (A.1),(A.2),(A.3) and orthogonality of Θ we have:

$$\dot{J}_i = -2\mathrm{tr}\big(\Theta^T\Theta C_{i0}\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\Theta\Delta\big) + 2\mathrm{tr}\big(\Theta^T \ \mathrm{diag}(\Theta C_{i0}\Theta^T)\Theta C_{i0}\Theta^T\Theta\Delta\big)$$

$$= 2\mathrm{tr}\Big(\Theta^T\big[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\big]\Theta\Delta\Big) = 2\mathrm{tr}\Big(\Theta^T\big[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\big]\dot{\Theta}\Big)$$

where $[A, B] = AB - BA$ is the matrix Lie Bracket. By definition of the Riemannian metric in equation (5.2) and the definition of gradient with respect to this metric we have:

$$\nabla J_i = 2\big(\Theta^T\big[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\big]\big)^T = -2\big[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\big]\Theta =$$

$$-2\Theta\big[\Theta^T\mathrm{diag}(\Theta C_{i0}\Theta^T)\Theta, C_{i0}\big]$$

Accordingly:

$$\nabla J = \sum_{i=1}^{N} J_i = -2\,\Theta\sum_{i=1}^{N}\big[\Theta^T\mathrm{diag}(\Theta C_{i0}\Theta^T)\Theta, C_{i0}\big] = -2\sum_{i=1}^{N}\big[\mathrm{diag}(\Theta C_{i0}\Theta^T), \Theta C_{i0}\Theta^T\big]\,\Theta$$

Then the gradient flow for minimization of $J(\Theta)$ will be:

$$\dot{\Theta} = -\frac{1}{2}\nabla J \tag{A.8}$$

which proves the second part of the theorem.

A.2 Proof of Theorem 5.2

From $C_j = \Theta C_{j0} \Theta^T$ and by (A.8) we have:

$$\dot{C}_j = \dot{\Theta} C_{j0} \Theta^T + \Theta C_{j0} \dot{\Theta}^T = \sum_{i=1}^{N} \Big(\big[\mathrm{diag}(\Theta C_{i0} \Theta^T), \Theta C_{i0} \Theta^T\big] \Theta C_{j0} \Theta^T -$$

$$\Theta C_{j0} \Theta^T \big[\mathrm{diag}(\Theta C_{i0} \Theta^T), \Theta C_{i0} \Theta^T\big] \Big) = \sum_{i=1}^{N} \Big[\big[\mathrm{diag}(\Theta C_{i0} \Theta^T), \Theta C_{i0} \Theta^T\big], \Theta C_{i0} \Theta^T \Big]$$

$$= \sum_{i=1}^{N} \Big[[\mathrm{diag}(C_i), C_i], C_i \Big] = \Big[\sum_{i=1}^{N} [\mathrm{diag}(C_i), C_i], C_j \Big]$$

In deriving the last expression we used the definition of the Lie Bracket.

A.3 Proof of Theorem 6.1

We name each single term in the summation as $J_j(B)$ that is:

$$J_1(B) = \sum_{i=1}^{N} J_i(B) = \sum_{i=1}^{N} \big\| BC_iB^T - \mathrm{diag}(BC_iB^T) \big\|_F^2 \tag{A.9}$$

then note that using identities (A.1), (A.2) and the symmetry of C_i's we have:

$$J_i = \mathrm{tr}\Big(\big(BC_iB^T - \mathrm{diag}(BC_iB^T)\big) \big(BC_iB^T - \mathrm{diag}(BC_iB^T)\big) \Big) = \mathrm{tr}(BC_iB^TBC_iB^T)$$

$$-2\mathrm{tr}(BC_iB^T\mathrm{diag}(BC_iB^T)) + \mathrm{tr}(\mathrm{diag}(BC_iB^T)\mathrm{diag}(BC_iB^T)) = \mathrm{tr}(BC_iB^TBC_iB^T)$$

$$-\mathrm{tr}\big(BC_iB^T\mathrm{diag}(BC_iB^T)\big)$$

Next we compute the time derivative of $J_i(B)$ in terms of the time derivative of B i.e. $\dot{B} \in T_B GL(n)$. Using the symmetries and the same identities we will have:

$$\dot{J}_i(B) = 4\mathrm{tr}(C_iB^TBC_iB^T\dot{B}) - 4\mathrm{tr}\big(C_iB^T\mathrm{diag}(BC_iB^T)\dot{B}\big) =$$

$$4\mathrm{tr}\Big(\big(C_iB^TBC_iB^T - C_iB^T\mathrm{diag}(BC_iB^T)\big)\dot{B} \Big) = \mathrm{tr}(X\dot{B}) \tag{A.10}$$

Now from the definition of the Natural Riemannian metric in equation (6.2) and the corresponding gradient we have that:

$$\dot{J}_i = \mathrm{tr}((B^T B)^{-1} \nabla J_i^T \dot{B})$$

hence from (A.10) we have that: $\nabla J_i = X^T B^T B$, therefore:

$$\nabla J_i = 4\big((BC_iB^T - \mathrm{diag}(BC_iB^T))BC_iB^T\big)B$$

and $\nabla J_1 = \sum_{i=1}^{N} \nabla J_i$ yields:

$$\nabla J_1(B) = 4\bigg(\sum_{i=1}^{N}\big(BC_iB^T - \mathrm{diag}\big(BC_iB^T\big)\big)BC_iB^T\bigg)B$$

which is the first statement of the theorem and the next statement is simply letting $\dot{B} = -\frac{1}{4}\nabla J_1(B)$.

BIBLIOGRAPHY

[Amari 1] S. Amari, "Differential-geometrical methods in statistics", Lecture notes in statistics, Springer-Verlag, Berlin, 1985 ,
[Amari 2]S.Amari, T.-P Chen, A. Chichoki, "Non-holonomic constraints in learning algorithms for blind source separation" , preprint, 1997.
[Brockett] R.W. Brockett, "Differential Geometry and the Design of Gradient Algorithms", Proceedings of Symposia in Pure Mathematics, Vol. 54, pp 69-92, American Mathematical Society, 1992.
[Bunse-Gerstner] A. Bunse-Gerstner, R. Byers and V. Mehrmann, "Numerical Methods For Simultaneous Diagonalization", SIAM Journal on Matrix Analysis and Applications, vol. 4, pp. 927-949, 1993.
[Calvo] M.P. Calvo, A. Iserles and A. Zanna, "Runge-Kutta methods for orthogonal and isospectral flows", Appl. Num. Maths 22 (1996).
[Cardoso1] J.F. Cardoso and A. Soulumiac, "Blind Beamforming For Non-Gauusian Signals", IEE-Proceedings, Vol.140, No 6, Dec 1993.
[Cardoso2] J.F Cardoso and B Laheld. "Equivariant adaptive source separation", IEEE Transactions on Signal Processing, vol. 44, no 12, pp. 3017-3030, Dec. 1996.
[Cardoso3] J.F. Cardoso, "Infomax and maximum likelihood for source separation", IEEE Letters on Signal Processing, vol. 4, no. 4, pp. 112-114, April, 1997.
[Comon1] P.Comon, "Independent Component Analysis, A New Concept?", Signal Processing, Vol.36, No 3, Special issue on High-order Statistics, April 1994.
[Cover] T.M. Cover and J.A. Thomas, "Elements of Information Theory", Wiley Inter-Science, 1991.
[De Lathauwer1] L. De Lathauwer, "Signal Processing Based on Multilinear Algebra", Ph.D dissertation, Katholiecke Univ. Leuven, Belgium, 1997.
[De Lathauwer2] L.De Lathauwer, P.Comon, B. De Moor, "A contrast-Based Independent Component Analysis without Second-Order Moments", Submitted to: 8th IEEE SP workshop on Statistical Signal and Array Processing, Corfu, Greece, 1996.
[Devore] R. Devore, A. Iserles and E. Sli, "Foundations of Computational Mathematics", Cambridge University Press, 2001.
[Golub], G.H. Golub, C.F. Van Loan, "Matrix Computations", Johns Hopkins Se-

ries in the Mathematical Sciences, 1996.
[Grellier] O. GRELLIER and P. COMON, "Blind Separation of Discrete Sources", IEEE Signal Processing Letters, 5(8):212–214,1998.
[Haykin] S. Haykin (Editor), "Unsupervised Adaptive Filtering, Vlume I, Blind Source Separation", Wiley Interscience, 2000.
[Helmke] U.Helmke and J.B. Moore, "Optimization and Dynamical Systems", Springer-Verlag, 1994.
[HOWE] R. Howe, "Very Basic Lie Theory", American Mathematical Monthly, pp 600-623, Nov 1983.
[Hyavarinen] A. Hyvarinen, J. Karhunen and E. Oja, "Independent Componenet Analysis", Wiley Inter-Science, 2001.
[Iserles] A. Iserles, "On Cayley-transform methods for the discretization of Lie-group equations", Found. Comp. Maths 1 (2001).
[Khalil] H.K.Khalil, "Nonlinear Systems", Prentice Hall, 2002.
[Mahony], R. Mahony and Ben Andrews, "Convergence of the Iterates Descent Methods for Analytic Cost Function", preprint
[Marcus] M. Marcus and H. Minc, "A Survey of Matrix Theory and Matrix Inequalities", Dover Publications, 1964.
[Marsden] J.E. Marsden and T.R Ratiu, "Introduction to Mechanics and Symmetry", Second Edition, Springer-Verlag, 1999.
[McCullagh] p. McCullagh, "Tensor methods in statistics", Chapman and Hall, 1987.
[Moreau] E.Moreau , "A generalization of joint-diagonalization criteria for source separation", IEEE Transactions on Signal Processing , vol. 49 , n 3 , 2001.
[Papoulis] A. Papoulis, S.U. Pillai,"Probability, random variables, and stochastic processes", McGraw-Hill, 2002.
[Pham1] D.T. Pham and J.F Cardoso, "Blind Separation of Instantaneous Mixtures of Non Stationary Sources", IEEE Transactions on Signal Processing, pp 1837-1848, vol 49, no 9, 2001.
[Pham2] D.T. Pham, "Joint Approximate Diagonalization of Positive Definite Hermitian Matrices", SIAM Journal of Matrix Analysis and Applications, Vol. 22, No. 4, pp. 136-1152.
[Porat] B. Porat, "Digital Processing of Random Signals", Prentice Hall, 1993.
[Stuart] A.M. Stuart and A.R. Humphries, "Dynamical Systems and Numerical Analysis", Cambridge University Press, 1996.
[Tits] A. Tits, "Notes for Optimal Control", Lecture notes for ENEE 664 course, University of Maryland, College Park, 1998.
[Yeredor] A.Yeredor,"Non-Orthogonal Joint Diagonalization in the Least-Squares Sense With Application in Blind Source Separation", IEEE Transactions on Signal Processing, Vol 50, No.7.July 2002.
[Zhang]L. Zhang, A. Cichocki and S. Amari, "Natural gradient algorithm for blind separation of overdetermined mixture with additive noise", IEEE Signal Process-

ing Letters, Vol. 6 , No. 11, 1999.
[Ziehe], A.Ziehe, P. Laskov, K. Muller, G. Nolte,"Linear Least-squares Algorithm for Joint Diagonalization", 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003), April 2003, Nara, Japan